From Small to Large Baseline Multiview Stereo: Dealing with Blur, Clutter and Occlusions

Qingxu Dou

A thesis submitted for the degree of Doctor of Philosophy

Heriot-Watt University
School of Engineering and Physical Sciences
Department of Electrical, Electronic & Computer Engineering

May 2011

Acknowledgements

First and foremost I would like to thank my supervisor Dr. Paolo Favaro with whom I have been fortunate to work during the last four years. It would have been impossible to finish this work without his guidance, patience, support and encouragement. His vast knowledge in the fields of computer vision and mathematics helped me build a solid foundation in computer vision and become a researcher in this area.

I am grateful to Professor Steve McLaughlin for hours of discussions about computer vision, how to start a PhD study and other random topics which helped me a lot during the course of my PhD.

I am grateful to Professor Chris Eilbeck, not only for his supervision when I was doing my MSc degree, but also for his friendship, encouragement and sound advices during these years.

I would like to thank all the other members in our group (Paolo Favaro, Tom Bishop, Manuel Martinello, Thoma Papadhimitri, Daniele Perrone) for the valuable discussions and advices in the group meetings and Journal Club.

I would like to thank all my friends and colleagues in Heriot-Watt University for their help. I am grateful to the Joint Institute in Signal and Image Processing for offering me the opportunity of doing this degree.

At last but not least, I would like to thank my family, especially my wife, for their love, support and unfailing faith in me.

Abstract

This thesis addresses the problem of reconstructing the three-dimensional (3D) digital model of a scene from a collection of two-dimensional (2D) images taken from it. To address this fundamental computer vision problem, we propose three algorithms. They are the main contributions of this thesis.

First, we solve multiview stereo with the off-axis aperture camera. This system has a very small baseline as images are captured from viewpoints close to each other. The key idea is to change the size or the 3D location of the aperture of the camera so as to extract selected portions of the scene. Our imaging model takes both defocus and stereo information into account and allows to solve shape reconstruction and image restoration in one go. The off-axis aperture camera can be used in a small-scale space where the camera motion is constrained by the surrounding environment, such as in 3D endoscopy.

Second, to solve multiview stereo with large baseline, we present a framework that poses the problem of recovering a 3D surface in the scene as a regularized minimal partition problem of a visibility function. The formulation is convex and hence guarantees that the solution converges to the global minimum. Our formulation is robust to view-varying extensive occlusions, clutter and image noise. At any stage during the estimation process the method does not rely on the visual hull, 2D silhouettes, approximate depth maps, or knowing

which views are dependent(i.e., overlapping) and which are independent(i.e., non overlapping). Furthermore, the degenerate solution, the null surface, is not included as a global solution in this formulation. One limitation of this algorithm is that its computation complexity grows with the number of views that we combine simultaneously. To address this limitation, we propose a third formulation. In this formulation, the visibility functions are integrated within a narrow band around the estimated surface by setting weights to each point along optical rays.

This thesis presents technical descriptions for each algorithm and detailed analyses to show how these algorithms improve existing reconstruction techniques.

Notation

a	scaler
$\mathbf{a}$	column vector
$\mathbf{b}^T$	transpose of $\mathbf{b}$
$\mathbf{C}$	camera centre or viewpoint
$\mathbf{P}$	3-D point
$\mathbf{p}$	2-D image point
P	projection matrix of a camera
$\mathbf{S}$	3-D surface
$\mathbf{V}$	3-D volume
N	number of photographs or cameras
$\mathbf{N}$	normal vector
v	distance from the image to the lens
v_i	distance from the image plane to the lens
f	focal length of a lens
u	distance fro an object to the lens
K	camera calibration matrix
R	rotation matrix
h	point spread function
$\overrightarrow{\mathbf{AB}}$	vector from $\mathbf{A}$ to $\mathbf{B}$
$\mathbb{R}^3$	3-D space
Ω	region in space
$\delta\Omega$	boundary of Ω

$\delta\phi$	diferentiation of function ϕ
ρ	probability of an event

Contents

List of Figures

List of Tables

Publications of the Candidate

- Q. Dou and P. Favaro Off-axis Aperture Camera: 3D Shape Reconstruction and Image Restoration. CVPR 2008, pages 1-7.
- Q. Dou and P. Favaro A Convex Multi-view Stereo Formulation with Robustness to Occlusions and Time-varying Clutter. ICVGIP 2010, pages 243-250.
- Q. Dou and P. Favaro A Occlusions and Clutter Robust Convex Formulation of Multiview Stereo. Going to be submitted to The Visual Computer.

Chapter 1

Introduction

1.1 Why Multiview Stereo?

3D reconstruction is the process of reconstructing the shape and appearance of a scene from a set of images of this scene. This is a classical problem in computer vision and needed in many fields such as robot navigation, virtual reality, augmented reality, tracking and so on. This thesis addresses this classical problem by solving multiview stereo and focuses on dealing with blur, clutter and occlusions depending on how images are collected.

The process of 3D reconstruction can be accomplished either by active or passive methods depending on which sensors are used. Active sensors like lidar scanners and laser triangulation sensors estimate distance by interfering directly with the reconstructed objects. Passive sensors record the radiance reflected or emitted by the objects' surface to infer 3D structure without directly interfering with it. Therefore, passive methods can be used when it is possible to control the scene illumination or when does not want the object sensing that it is being probed. Furthermore, systems based on active sensors are typically more expensive. As a passive system, we consider a camera with an image sensor in the visible spectrum.

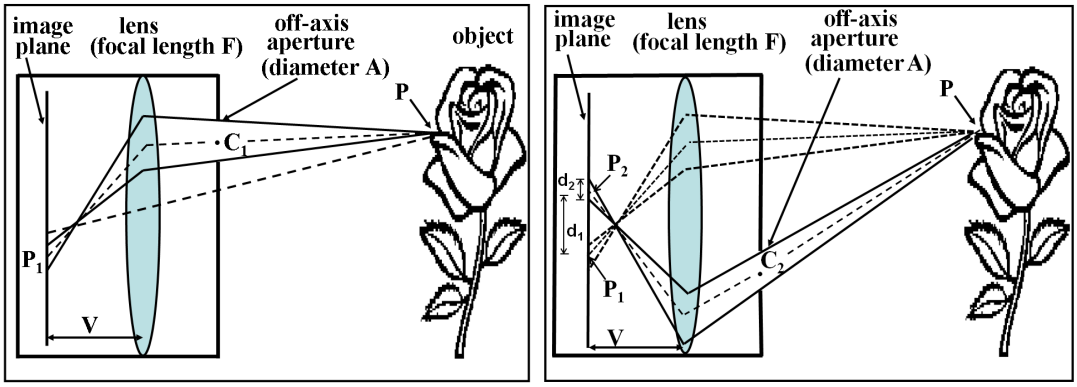


Figure 1.1: The geometry and defocus of the off-axis aperture camera. For a surface point P which is not in focus, its image is a blurred disc. When the aperture center is at C_1, the center of the blurred disc is at P_1 (left image), if the aperture center is rotated to C_2, the center of the blurred disc is changed to P_2 (right image). The distance between P_1 and P_2 and the diameter of the blurred disc are also shown in the right image with d_1 and d_2. It can be seen that they are comparable to each other. This means that compared to the displacement of the projection, the defocus effect can not be neglected. It can be also seen that the diameter of the blurred disc is related to the size of the aperture, the distance between the aperture and the lens along the direction of the optical axis and the distance from the object to the camera.

There are several passive methods to reconstruct the 3D shape of a scene which include shape from shading, shape from motion, shape from defocus, stereo, multiview stereo and so on. Compared to other techniques, multiview stereo has the following advantages: First, dense depth maps can be obtained for each viewpoint; Second, depth resolution and texture can be enhanced via viewpoint fusion; Third, the shape of a full scene can be reconstructed.

1.2 Effects of the Baseline Scale in Multiview Stereo

An excellent survey on multiview stereo is given by Seitz et al. in [146]. As shown in that survey most methods work with the images captured with a relatively large baseline. However, when the baseline i. e. the distance between cameras changes

from very small to large, the differences of the constraints in multiview stereo are significant. For example, when the baseline between cameras is very small, occlusions do not pose a serious problem but defocus becomes more significant. On the contrary, when the baseline between cameras is large, occlusions become more relevant. In this work, we focus on dealing with blur when we solve multiview stereo with very small baseline and dealing with occlusions when we solve multiview stereo with large baseline.

For solving multiview stereo with a very small baseline, we build our work based on an off-axis aperture camera model as depicted in Figure 1.1. By rotating the aperture around the optical axis or changing the distance between the aperture and the lens along the optical axis or the size of the aperture, different views of the scene are selected. The illumination shows that the same point on the surface of an object will be projected at different locations and with different amount of blur on the image plane when the object is not in focus. When the baseline is very small, compared to the difference of the locations of the projections of the object, the amount of blur is significant. In our formulation we present a novel framework which combines stereo and defocus to obtain the 3D surface reconstruction and image restoration in one go.

In solving multiview stereo with a large baseline, occlusion is an unavoidable problem. The occlusions we meet in multiview stereo can be divided into two classes: self-occlusions and occlusions from other objects. Figure 1.2 shows examples of both types of occlusions. In each view, some parts of the statue are self-occluded. In the left image of Figure 1.2, some parts of the statue are occluded by other objects, the tourists. If the tourists were walking around when these images were taken, they would appear in different locations relative to the statue in different views. In this thesis we call such occlusions view-varying occlusions. In the right image in Figure 1.2, the texture of the base of the statue is similar to that of the wall behind it. When the foreground and background have similar texture, we say that there is clutter. The clutter also could change with

Figure 1.2: The occlusions and clutter. It can be seen that in each view some parts of the statue are self-occluded. In the left image, some parts of the statue are occluded by the tourists. The tourists are the occlusions from other objects with respect to the statue. The texture of the base of the statue is similar to that of the wall behind it, so there is clutter in both views.

the viewpoint and hence we call it view-varying clutter.

Most methods for multiview stereo are robust to self-occlusions and some of them are robust to occlusions from other objects [58, 61, 66, 160]. However none is robust to occlusions and is formulated as a convex problem. Convexity is desirable to guarantee convergence to a global minimum. There exists work in multiview stereo using a convex formulation such as [94, 189] but without robustness to the occlusions.

1.3 Contributions of this Work

The contributions of this work are as follows: First, to solve multiview stereo with a very small baseline, such as in the off-axis aperture imaging device, we propose a formulation which can achieve shape reconstruction and image restoration in one go by combining stereo and defocus cues. The off-axis camera is a versatile imaging device. We can use it to capture multiple images by rotating the aperture around the optical axis, changing the size of the aperture or changing the distance from the aperture to the lens along the optical axis. Nobody has modeled a camera in this way before. We introduce a novel image formation model and also introduce the concept of scene space rectification to simplify the analysis.

Second, to solve multiview stereo with a large baseline, we propose a formulation which is not only convex but also very robust to view-varying occlusions and clutter. This is the first time that the convexity and robustness to occlusions and clutter are formulated in one framework. As mentioned earlier, this framework does not relay on the visual hull, 2D silhouettes, or approximate depth maps. Another very important feature is that we do not require knowing which views are dependent (i.e., overlapping) and which views are independent(i.e., non overlapping). This information is an essential ingredient in several methods to render the formulations convex. The degenerate solution, the null surface, is not included as a global solution in this formulation.

Third, we also propose another formulation for solving multiview stereo with a large baseline. The contributions in this case are more technical. We introduce a weighting scheme to ensure that computations are carried out only where they are needed, i.e. around the surface. Compared to the previous formulation, this formulation maintains the same robustness to view-varying occlusions and clutter, but integrates information from different views in a straightforward manner and is more computationally efficient.

1.4 The Structure of this Thesis

Chapter 2 provides a review on the methods for dense reconstruction by solving multiview stereo. They are divided into four classes: Voxel-based 3D volumetric approaches, the algorithms that compute and merge depth maps, partial differential equation(PDE) based surface evolution methods and techniques to reconstruct 3D surfaces by region growing. Descriptions are given to each class in the subsections.

Chapter 3 focuses on the image formation models. It starts with the trajectory of lights in image formation procedure, then presents the pinhole camera model and finally concludes with the camera model with a finite aperture. In multi-view stereo, most proposed methods work with calibrated images. The camera pose is often described by the camera projection matrix which contains the intrinsic and extrinsic parameters of the corresponding camera. The process of camera calibration to determine the values of the intrinsic and extrinsic camera parameters is also described briefly in this chapter.

Chapter 4 introduces the level set method devised by Osher and Sethian [132] first and then continues with the numerical schemes for solving the corresponding formulations in level set methods. The numerical schemes include the *upwind*

scheme, the *essentially nonoscillatory(ENO)* schemes and the *weighted essentially nonoscillatory(WENO)* scheme. The Chan-Vese algorithm [31], which is connected very closely to the third algorithm proposed in this thesis, is reviewed briefly in this chapter. We tested the effects of some parameters in the Chan-Vese algorithm on the behavior of the interface and the convergence to the true segmentation and compared the accuracy of different numerical schemes. The corresponding experimental results are shown with 2D and 3D segmentations.

Chapter 5 introduces the formulation for solving multiview stereo with a very small baseline. First the geometry of the proposed off-axis aperture camera model is presented. The image formation model describes both stereo and defocus. To simplify analysis we use *space warping*. A gradient flow minimization algorithm is then introduced and tested on synthetic and real data.

Chapter 6 introduces two methods for solving multiview stereo with a large baseline.

In the first part of this chapter, we present a view-varying occlusions and clutter robust convex formulation by posing the task of reconstructing the 3D surface as a partition problem. First, the energy formulation is explained in detail and compared with other works. Next, we show how to compute the depth map with respect to a central camera. By comparing a few schemes for integrating the depth map representations, we show the advantages of the proposed integration scheme. After the numerical implementation of the proposed formulation is presented, the first part of this chapter is terminated by displaying the experimental results.

In the second part of this chapter, we introduce a second formulation for solving multiview stereo with a large baseline based on integrating the depth map representations within a narrow band around the approximated surface.

Chapter 7 summarizes this thesis. It starts with a summary of this thesis and then continues with the main contributions of this work. This chapter is terminated with the future work we are going to do based on the work displayed in this thesis.

Chapter 2

A Review of Multiview Stereo

2.1 Introduction

There is a large body of work on multiview stereo(MVS) algorithms for dense shape reconstruction (see [146] for an excellent recent survey). Among the most successful methods, are those based on shape from silhouette, which obtain an estimation of the 3D surfaces from binary object/background segmentations of each view [165, 176]. These methods are known to be robust and computationally efficient, but cannot reconstruct all concavities. Other popular approaches are those based on space carving, where voxels that do not correspond to pixels that are photoconsistent are removed [98]. These methods have the limitation that regularization is not enforced and reconstructions are often noisy. Solutions that incorporate regularization have also been proposed. In [51, 159, 186] a deformable model is updated in a variational minimization scheme until a certain consistency criterion is satisfied. This approach allows to combine a data fidelity term on the unknown surface, which measures how well the solution is consistent with the data, with a regularization term, which constrains the solution to be smooth. Although these methods achieve a higher robustness to image noise, they inherently define the empty set as a global optimum and typically depend

on the initialization. To compensate for such limitations, the volumetric graph cuts method proposed in [177] incorporates a ballooning term. In addition, it is a global minimization framework which improves the robustness considerably and loosens the requirement of an initial guess. However the globality of the graph cuts is guaranteed only in a discrete sense and increasing the space resolution will also increase the computations dramatically. Other very effective techniques include methods which merge depth maps obtained from small groups of neighboring views [27, 79] and methods that reconstruct the final surface by growing from a sparse set of "seed patches" from image features [61, 73].

According to the taxonomy in [146], the MVS algorithms can be divided into four classes: voxel-based 3D volumetric approaches [83, 98, 145, 150, 155, 170, 177, 178], partial differential equations (PDEs) based surface evolution methods [49, 52, 64, 93, 136, 190], the algorithms that compute and merge depth maps [27, 65, 120, 160, 169, 189], and the techniques to reconstruct the 3D surfaces by region growing [61, 66, 73]. A description to each class will be given in the following sections.

2.2 Volumetric Scene Reconstruction

Volumetric data representations date back to the early 70's of the last century in the context of 3D medical imaging [70]. At the beginning there was a problem in storage space and computational speed, but with the dramatic improvement of technology, these representations have been becoming a practical and important tool in the field of computer vision. The earliest attempts at volumetric reconstruction from photographs are those that approximate the visual hull [12, 28, 55, 102, 116] of the imaged objects. These methods are robust and efficient but with some shortcomings. First, most of these methods are based on 2D silhouette images, which are binary images indicating whether a pixel is the projection of a surface point or projected from the background. The silhouette

images can be used to generate visual hulls. The visual hull of an object is an approximation of it, which can generate the same silhouette images as the object itself from any viewpoint. In practice, an approximated visual hull of a scene can be obtained by backprojecting the silhouette images into space and intersecting the generated 3D cones. For the images taken in a natural environment, especially in a cluttered scene, it is not easy or even possible to segment the foreground from the background. Without silhouette images, there is no way to generate the visual hull. It is a harsh constraint for the methods which reconstruct the 3D shape based on the visual hull. Second, some concave parts can not be reconstructed with these methods no matter how many views are being used. Some parts of the reconstructed visual hull surface can be quite far from the true surface. Space carving methods are another group of approaches working with volumetric data representations. In this group of approaches, voxels that do not correspond to pixels that are photoconsistent are removed [39, 145]. These methods introduce a bias towards maximal photoconsistent shapes and do not enforce smoothness, which often result in rather noisy reconstructions. Volumetric graph cuts formulate the 3D reconstruction problem as an energy minimization problem of a Markov Random Field [83, 106, 177, 178]. In this group of methods, usually the cost functionals consist of two terms: foreground/background cost and discontinuity cost. The smoothing constraint can be applied via the discontinuity cost term. In the following subsections, a description on visibility is given first and then the methods in each class are described briefly.

2.2.1 Visibility

As an important topic in computer graphics, computational geometry and scene analysis, the notion of visibility is very important in computer vision, especially in multiview stereo. Many visibility algorithms have been proposed in the literature of computer graphics and several surveys of visibility methods have been

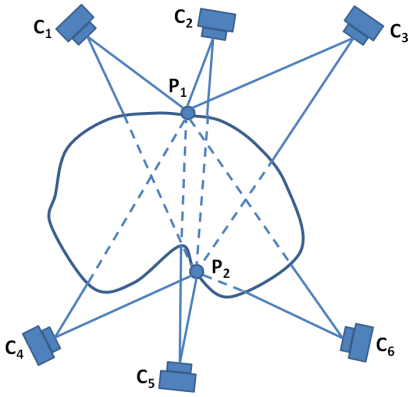


Figure 2.1: A 2D example of visibility. From this camera configuration it can be seen that point P_1 can be seen by cameras C_1, C_2 and C_3 but can not be seen by cameras C_4, C_5 and C_6 and point P_2 can only be seen by cameras C_4 and C_5.

published [15, 37, 46, 69, 167].

In multiview stereo, if a camera can see a certain patch on the surface of a scene it is quite possible that another camera can not see it, especially when the baseline is large (see Figure 2.1 for a 2D example). To reconstruct this patch, the cameras which can not see it can not provide any useful information and such wrong cameras usually lead to incorrect reconstructions. Knowing what surfaces can be "seen" by a certain camera amounts to knowing the visibility mapping. Hence, the visibility and the geometry of a scene are directly related. Given the geometry it is straightforward to obtain the visibility; vice versa, given the visibility, it is possible to obtain the geometry. Several strategies to approximate the visibility before the reconstructing the geometry have been proposed and some alternate between evaluating the two qualities.

Esteban and Schmitt [49] treat occlusion as an additional source of image noise. For a certain pixel, the intersection of the optic ray passing through the optical center and this pixel with the visual hull is quantized into different depth levels. Individual correlations are computed with different images to obtain several correlation curves. The criterion used to find the best candidate depth along the optic rays is based on the correlation values and also on the coherence be-

tween the correlation curves. If the four nearest neighbor cameras are used to calculated the correlation curves, up to 2 bad correlation curves due to highlights, occlusions or non-coherent textures occurring in the corresponding images can be tolerated.

Some methods use an initial approximation of the true surface such as the visual hull to estimate the visibility [93, 178]. The problem with these methods is that the visual hull is not always available and, in addition, at the concavity parts, the surface of the visual hull could be quite far from the true surface, leading to wrong visibility approximation and then poor reconstruction.

In the variational 3D reconstruction methods, the surface obtained at a certain iteration is used to approximate the visibility for the next iteration. With the improvement of the reconstruction, the approximated visibility may converge to the true one [90].

A probabilistic framework for estimating the visibility is proposed in [79]. This framework is built upon the assumptions that the probability distributions of the true depths and the measured ones are independent and the probability distribution of a sensor measurement given the scene depths and all other measurements only depends on the surface depth it is measuring. The proposed visibility criterion is related very closely to the depth-map fusion method proposed by Curless and Levoy [40].

2.2.2 Volumetric Visual Hulls

A silhouette image is a binary image, with the value at a pixel indicating whether this pixel is the projection of a point on the surface of the object or the projection of the background. The silhouette images can be obtained by segmentation algorithms [22, 164] or blue-screen techniques. In practice, to facilitate the extraction of the silhouettes, the images are usually captured in a controlled environment with black or white background. Then, the silhouettes can be obtained by thresh-

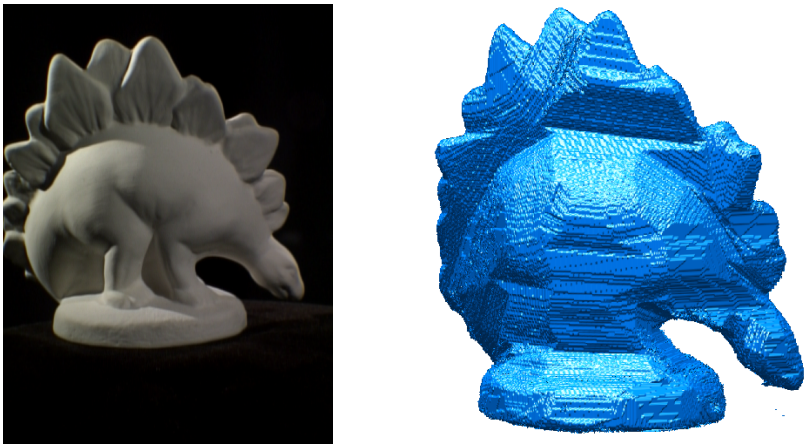


Figure 2.2: The reconstructed visual hull of Dinosaur dataset. Left: one of the input images (download from:http://vision.middlebury.edu/mview/data/. Right: the reconstructed visual hull.

olding the images.

The visual hull of an object is the best approximation to the object which can be obtained by using an infinite number of silhouettes captured from all view points outside the convex hull of the object. If a viewing region is defined as a set of points in space from which silhouettes of an object are seen, a formal definition of visual hull with respect to a certain viewing region is given by Laurentini [103] in the following way:

Definition *(Visual hull). The visual hull of an object* $\boldsymbol{V}$ *with respect to a viewing region* Ω*, denoted* $VH(\boldsymbol{V}, \Omega)$*, is a volume in space such that for each point* $\boldsymbol{P} \in VH(\boldsymbol{V}, \Omega)$ *and each viewpoint* $\boldsymbol{C} \in \Omega$*, the half-line from* $\boldsymbol{C}$ *through* $\boldsymbol{P}$ *contains at least one point of* $\boldsymbol{V}$*.*

In most cases, it is impossible to obtain an accurate visual hull from finite silhouette images. But in this situation, it is still preferred to use a *visual hull* instead of an *approximated visual hull*. So in this thesis, when we talk about a visual hull it is usually an approximated one.

Many algorithms have been proposed for reconstructing 3D models from a group of silhouette images [6, 12, 34, 103, 104, 105, 116, 117, 127, 128, 137, 153,

157, 168] and there are many other 3D reconstruction algorithms which work depending on visual hull as an initial guess [49, 59, 83, 93, 158, 170, 188]. In practice, the viewing region only contains a finite number of viewpoints, leading to an approximation of the visual hull depending on the number of views, the location of the viewpoints and the complexity of the object.

The key step in visual hull techniques is the intersection test. Some methods back-project the silhouettes and create an explicit set of cones that are then intersected in 3D [128, 157]. Some methods discretize the bounding volume, which contains the entire scene, into voxels and then project each voxel into all silhouette images. The voxels whose projections are contained in every silhouette belong to the visual hull [168]. To make the algorithms more efficient, most methods use an octree representation and test voxels in a coarse-to-fine hierarchy. The shape-from-silhouette problem has also been formulated as an optimization problem [153] where the shape is reconstructed by computing a global minimum of an energy function.

As shown in Figure 2.2, the visual hull techniques can not reconstruct all the concavities of the surface of the objects. For example, consider the part between the legs of the dinosaur. In this example, 72 silhouette images are used to generated the visual hull. In fact no matter how many silhouette images are used the concave part between the legs can not be reconstructed properly. Some points on the surface of the reconstructed visual hull can be quite far from the true surface.

Despite the inaccuracy of the 3D reconstruction, shape from silhouettes have been used successfully in many applications, such as tracking [5, 19], virtual reality [121], real-time human motion modeling [34], constructing a light field [68] and so on.

2.2.3 Volumetric Photo Hulls

If the background identification can not be performed accurately [152] or there is no background pixel in the images, the visual hull reconstruction method will fail. However, if the input images are grayscale or color ones, the additional photometric information can be used to improve the 3D reconstruction process.

Kutulakos and Seitz [98] developed a theory for reconstructing arbitrarily-shaped scenes from arbitrarily-positioned cameras by formulating shape recovery as a constraint satisfaction problem. They showed that any set of photographs of a rigid scene defines a collection of picture constraints that are satisfied by every scene projecting to those photographs. They characterized the set of all 3D shapes that satisfy these constraints and designed a practical reconstruction algorithm, called *Space Carving*, based on the underlying theory. Furthermore, they showed how to use the Space Carving algorithm to compute, from N photographs of an unknown scene, a maximal shape that encloses the set of all photo-consistent reconstructions, the *photo hull*.

The key step of the Space Carving algorithm is the photo-consistency test to determine whether or not a voxel should be removed. Three definitions about photo-consistency are given in [98], they are *Point Photo-Consistency*, *Shape-Radiance Photo-Consistency* and *Shape Photo-Consistency*.

Definition *(Point Photo-Consistency). Let* $\mathbf{V}$ *be an arbitrary subset of* $\mathbb{R}^3$*. A point* $\mathbf{P} \in \mathbf{V}$ *that is visible from* $\mathbf{C}$ *is photo-consistent with the photograph at* $\mathbf{C}$ *if (1)* $\mathbf{P}$ *does not project to a background pixel, and (2) the color at* $\mathbf{P}$*'s projection is equal to* rad_{PC} *(the radiance from* $\mathbf{P}$ *to* $\mathbf{C}$*). If* $\mathbf{P}$ *is not visible from* $\mathbf{C}$*, it is trivially photo-consistent with the photograph at* $\mathbf{C}$*.*

Definition *(Shape-Radiance Photo-Consistency). A shape-radiance scene description is photo-consistent with the photograph at* $\mathbf{C}$ *if all points visible from* $\mathbf{C}$ *are photo-consistent and every non-background pixel is the projection of a point in the shape.*

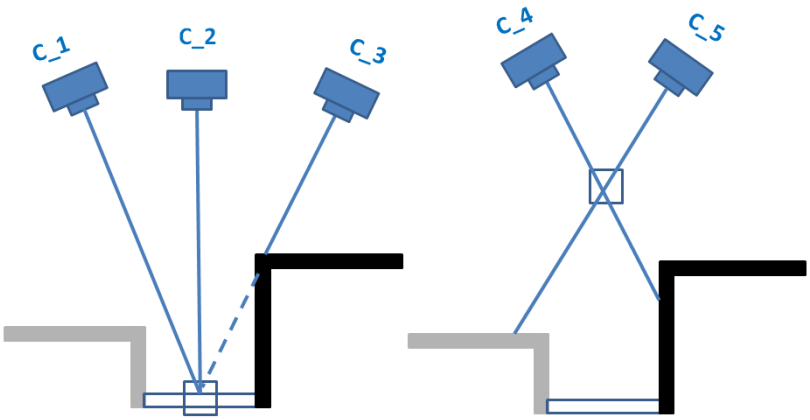


Figure 2.3: Color consistency check. On the left, Cameras C_1 and C_2 see consistent colors at a point on the surface but this point are occluded from camera C_3. On the right, cameras C_4 and C_5 see the inconsistent colors at a point not on the surface.

Definition *(Shape Photo-Consistency). A shape* $\mathbf{V}$ *is photo-consistent with a set of photographs if there is an assignment of radiance functions to the visible points of* $\mathbf{V}$ *that makes the resulting shape-radiance description photo-consistent with all photographs.*

Generally speaking, with respect to a group of N photographs, more than one shapes will be photo-consistent with them. Is there a member in this class which subsumes all the other members in this class? The answer is given by the *Photo Hull Theorem* in [98] as follows:

Theorem 2.2.1 (Photo Hull Theorem). *Let* $\mathbf{V}$ *be an arbitrary subset of* $\mathbb{R}^3$*. If* $\mathbf{V}^\star$ *is the union of all photo-consistent shapes in* $\mathbf{V}$*, and then every point on the surface of* $\mathbf{V}^\star$ *is photo-consistent.* $\mathbf{V}^\star$ *is called the photo hull. If it is closed, it is a photo-consistent shape.*

The theorem explicitly expresses the relation between the photo hull and all other possible 3D photo-consistent shapes with respect to a group of photographs: the theorem guarantees that every such photo-consistent shape is a subset of the photo hull.

The algorithm of Space Carving, which has been proven correct, provides a

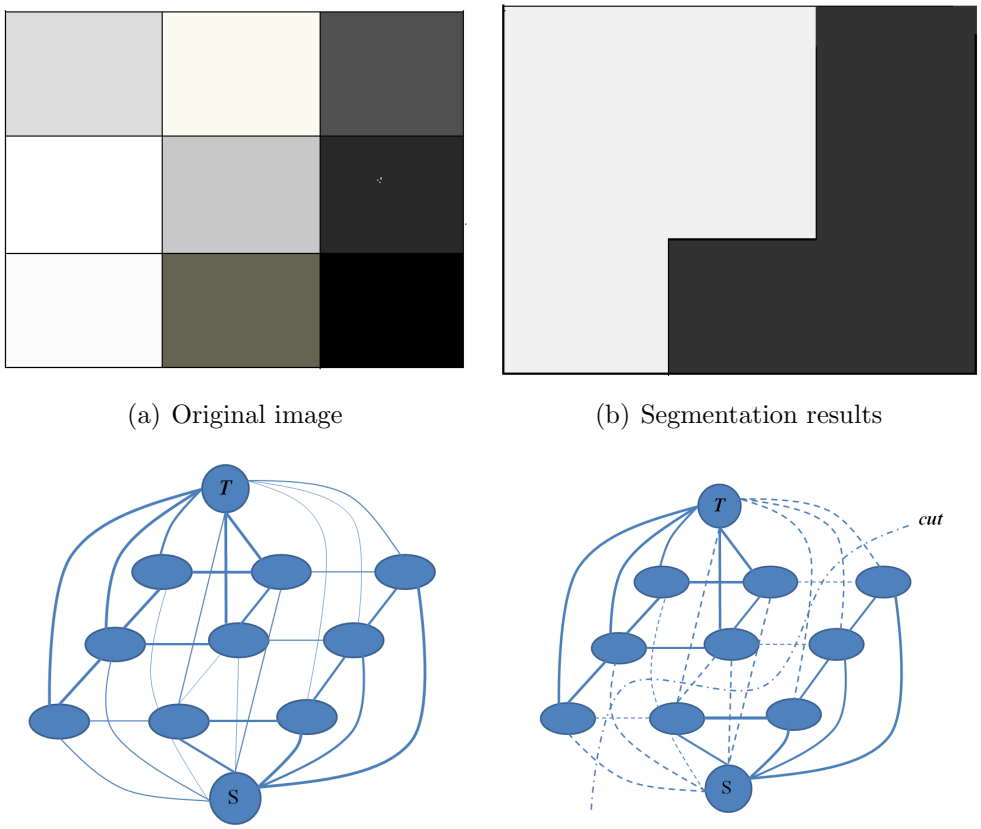
way to calculate the photo hulls [98]. Given an initial volume $\mathbf{V}$ that contains the scene, the algorithm proceeds by iteratively removing portions of that volume until it converges to the photo hull, $\mathbf{V}^{\star}$. 3D reconstruction based on photoconsistency requires camera parameters for each used view and a model for the object surface reflectance. Furthermore, a very important issue in this method is the criterion for consistency check. Most of the consistency criteria require a threshold from the user. Shape from photoconsistency methods often use the color consistency constraint, which is introduced by Seitz et al [145], to distinguish object surface voxels from other voxels. In addition, the voxel visibility problem has to be well addressed since the color consistency check for a voxel requires the set of images from which the voxel is visible. See Figure 2.3 as an example.

There are many variants of Space Carving [14, 47, 56, 97, 139, 151, 191]. Zeng et al. [191] introduce a local prior as a new interpretation of the smoothness assumption. This new prior leads to an effective implementation which combines advantages of space carving and graph cuts. Kutulakos [97] proposed a new multiview reconstruction algorithm, approximate N-view stereo, which is robust to noise and inaccurate calibration. A probabilistic framework to infer the photo hull distribution from a group of photos is introduced by Bhotika et al. in [14].

2.2.4 Volumetric Graph Cuts

One major limitation of the photo hull reconstruction methods is that a regularization is not enforced and reconstructions are often noisy. But in energy minimization formulations it is easy to incorporate a regularization term. Since the theory of graph cuts was first applied in computer vision in the paper by Greig et al. [71], many powerful energy minimization algorithms have been developed based on graph cuts [23, 25, 26, 83, 84, 95, 106, 138, 178].

Given a weighted graph with two distinguished terminal vertices called the source S and sink T, vertices can be partitioned into two disjoint sets, each of



(a) Original image (b) Segmentation results

(c) Graph built on the image (d) Minimum cut

Figure 2.4: An example of 2D segmentation with graph cuts. The cost of each edge is reflected by the edge's thickness.

which has the source S and sink T as a member. Each such partition is a cut of the weighted graph. The cost of the cut is the sum of the weights of the edges that join vertices that have been assigned to different sets. The minimum cut problem is to find the cut with the smallest cost. Due to the theorem given by Ford and Fulkerson [54], this problem can be solved by computing the maximum flow between the terminals in low-order polynomial time. In practice, the running time can be reduced to nearly linear for graphs with many short paths between the source and the sink [96]. A simple 2D segmentation example of graph cuts is shown in Figure 2.4.

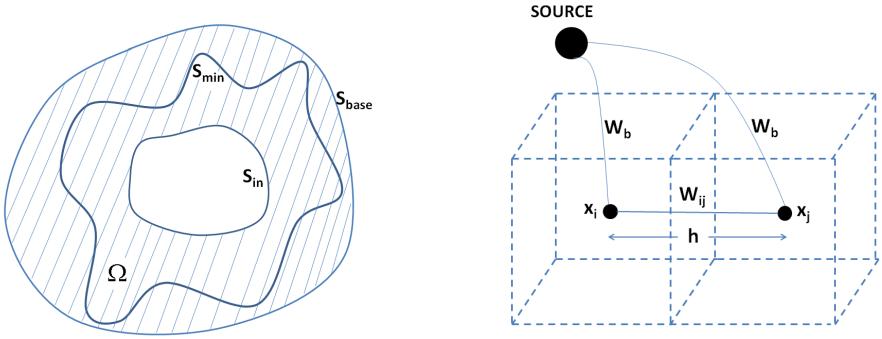


Figure 2.5: Surface geometry and flow graph construction. Left: a 2D slice of the volume showing the relative locations of the base surface $\mathbf{S}_{base}$, the inside boundary surface $\mathbf{S}_{in}$ and the scene surface $\mathbf{S}_{min}$ which represents the minimum cut in the flow graph. The region between $\mathbf{S}_{base}$ and $\mathbf{S}_{in}$ is represented with Ω. Right: two voxels in Ω and the corresponding nodes in the graph. Each voxel is connected to its neighbors and the source.

Based on the Riemannian minimal surface idea given in [24], Vogiatzis et al. [178, 179] provide a novel formulation for the multiview scene reconstruction problem, which combines a volumetric scene representation and a computationally tractable global optimization using graph cuts. To handle the occlusion and define connections to the source and sink, a base surface is used in the formulation in [178]. A choice for the base surface is the visual hull of the scene, which can not only be used to infer the occlusions but also to approximate the topology of the scene. In addition, an inner boundary surface is defined based on the base surface and a signed distance function that guarantees that the minimal surface lies between the base and the inner boundary surfaces. The relationship of the base, the inner boundary and the minimal surfaces is shown by a 2D slice of the space in Figure 2.5 (left). In this formulation, the graph vertices consist of all voxels whose center are between the base and inner boundary surfaces. The weight of the edge joining the two corresponding vertices on the graph is defined based on a photo-consistency measure and the size of the voxels. The configuration of the graph is shown in Figure 2.5 (right).

Although the potential of graph cuts is attractive to many researchers, there

are some shortcomings coming along with these techniques. Firstly, the globality of the solution obtained from graph cuts is only in a discrete sense and the metrication errors can not be avoided. Although such inaccuracies can be alleviated by increasing the space resolution, a finer space resolution can increase the computation load dramatically. Secondly, it is not practical to build huge graph structures since this entails considerable memory requirements. The possible remedy of using an adaptive multi-resolution scheme can not give any guarantee about the globality of the computed solution.

2.3 Reconstruction Based on Surface Evolution

Surface evolution is a technique used in many application domains such as material science, fluid mechanics and combustion. It is also a popular tool in computer vision. Numerous surface deformation schemes have been proposed over the last decades [49, 51, 52, 64, 136, 190]. They roughly fall into two main categories depending on the representation which is considered for surfaces: Lagrangian or Eulerian.

2.3.1 Lagrangian vs. Eulerian Surface Deformations

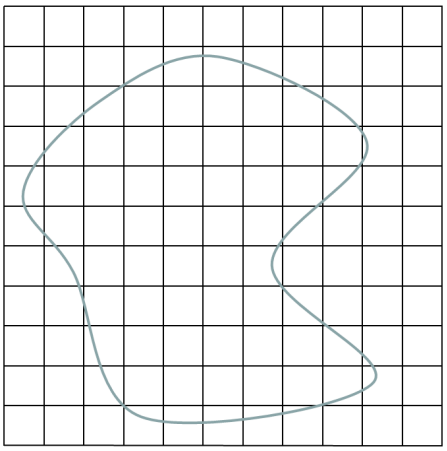
In Lagrangian methods, the surfaces are represented explicitly when they are deformed over time. There are numerous advantages in such representations, such as adaptive resolution, compact representation and the ability to handle non-geometric properties over the surface. But at the same time, there are two serious issues when an explicitly represented surface is evolved, namely self-intersections and topology changes, which make them difficult to use in many practical scenarios. Many efforts have been devoted to tackle these two issues. McInerney and Terzopoulos [119] introduced topology adaptive deformable curves and meshes, but this method is only suitable for certain motions of the surfaces. Pons and

Boissonat [135] proposed a method to evolve the mesh based on a restricted 3D Delaunay triangulation, but this method needs a strong assumption, i.e., the input mesh is sufficiently dense such that the Delaunay triangulation will not considerably change its layout. Some methods merge two surface boundaries when they are facing each other and closer than a threshold and split surface boundaries when they are back to back to each other [100], but in these methods the merges and splits are done in a heuristic sense. It is not easy to build a principle to fulfill this procedure. As a consequence, the Lagrangian methods are often neglected in favor of implicit representations which provide practical solutions to such issues.

Eulerian methods formulate the evolution problem as time variation over sampled spaces, most typically fixed grids. In such a formulation, the interface, which is a curve in 2D and a surface in 3D, is embedded in a higher dimensional space and represented implicitly. Two groups of the most successful methods in this category are the volume tracking methods [48, 129] and the level set methods [4, 52, 131, 132].

The volume tracking methods were introduced in the early 1970s [41, 129]. The basic idea of the volume tracking methods is similar to each other. That is for each cell in a grid (usually fixed), a value is assigned to it based on the fraction of that cell containing the material inside the interface. As given in Figure 2.6, the interface is a closed curve. A value of unity is assigned to those cells completely inside this curve, a value of zero to those completely outside, and a fraction between 0 and 1 to those cells that straddle the interface, based on the amount of the cell inside of the front. Then the interface is constructed solely based on these "cell fractions".

The difference between these methods lies in which geometry is chosen to reconstruct the interface. For example, DeBar's method [41] uses a piecewise linear approximation and more elaborate reconstruction techniques use pitched slopes and curved surfaces [35, 82, 101]. The resulting accuracy depends on the



(a) The grids and the interface

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.3	0.6	0.6	0.4	0.1	0	0	0
0	0.1	0.7	1	1	1	1	0.95	0.4	0	0
0.1	0.8	1	1	1	1	1	1	0.95	0.05	0
0.4	1	1	1	1	1	1	0.99	0.4	0	0
0.2	0.9	1	1	1	1	0.99	0.4	0	0	0
0	0.3	1	1	1	1	0.8	0	0	0	0
0	0.05	0.9	1	1	1	0.99	0.5	0.01	0	0
0	0	0.7	1	1	1	1	1	0.7	0.05	0
0	0	0.3	0.99	1	1	1	0.99	0.8	0.15	0
0	0	0	0.2	0.4	0.3	0.2	0.1	0	0	0

(b) The fractions for each cell

Figure 2.6: Volume-of-fluid method. A value is assigned to each cell in the grid based on the amount of the cell inside the interface. For example, a cell is assigned 1 if it is completely inside of the interface and 0 if it is completely outside. A fraction between 0 and 1 will be assigned to those cells that straddle the interface.

sophistication of the reconstruction and the "advection sweeps" which advance the material.

The volume tracking methods avoid some problems happening in Lagrangian methods such as topological change and Lagrangian time step. They are quite powerful. However there are some drawbacks: first, since the results are noticeably dependent on the underlying orientation of the grid, it is problematic to evolve under complex speed functions, especially in the presence of directional velocity fields. Second, since the approximation to the front through volume fractions is relatively crude, the techniques are inaccurate, and so are estimates of the intrinsic geometric properties, such as curvature and normal direction. Conversely, in level set methods, the level set equations can be accurately approximated by computational schemes which exploit techniques borrowed from the numerical solutions of hyperbolic conservation laws. More complex speed functions, such as those including curvature, are naturally framed in level set problems. The intrinsic geometric properties, such as the normal vector and the

curvature at a point of the front, can be easily determined. So with the development of level set methods, they are getting more and more popular in computer vision community. The details of these methods will be described in Chapter 4.

2.3.2 Variational Methods in Multiview Stereo

2.3.2.1 Variational Principles and the Euler-Lagrange Equation

Calculus of variations [87] is a field of mathematics that deals with functionals, as opposed to ordinary calculus which deals with functions. A functional is a real-valued function on a vector space, usually of functions. In practice, a functional involved in calculus of variations is built according to a certain variational principle for solving some physics problems. For example, to answer this question: What is the shape of a chain suspended at both ends? The functional built for solving this problem is based on the variational principle that the shape must minimize the gravitational potential energy.

The final purpose of solving a problem of calculus of variations is to find the extremal functions that make the functional attain a maximum or minimum value - or stationary functions - those where the rate of change of the functional is precisely zero. To fulfill this purpose, the key result to use in Calculus of Variations is the Euler-Lagrange equation, which is a differential equation whose solutions are the functions for which a given functional is stationary. The Euler-Lagrange equation is built upon two fundamental theorems:

Theorem 2.3.1 *Let the real-valued functional ϕ have a Gateaux differential $\delta\phi(\mathbf{x};\mathbf{a})$ on a vector space X. A necessary condition for ϕ to have an extreme at $\mathbf{x}_0 \in X$ is that $\delta\phi(\mathbf{x}_0;\mathbf{a}) = 0$ for all $\mathbf{a} \in X$, where Gateaux differential $\delta f(\mathbf{x};\mathbf{a})$ is defined as: $\delta\phi(\mathbf{x};\mathbf{a}) = lim_{\alpha\to 0}\frac{1}{\alpha}[\phi(\mathbf{x}+\alpha\mathbf{a}) - \phi(x)]$ and α is an arbitrary number.*

Theorem 2.3.2 *If $\varphi(t)$ is continuous on $[t_1, t_2]$ and $\int_{t_1}^{t_2} \varphi(t)\psi(t)dt = 0$ for every $\psi \in D[t_1, t_2]$ with $\psi(t_1) = \psi(t_2) = 0$, then $\varphi(t) \equiv 0$ on $[t_1, t_2]$, where $D[t_1, t_2]$ is*

the set of all real-valued functions defined on $[t_1, t_2]$.

Based on the above two theorems, the Euler-Lagrange equation can be derived. For example, to minimize the energy with the following form (single function of several variables):

$$E(\phi) = \int_{\Omega} \Phi(x_1, \cdots, x_n, \phi, \phi_{x_1}, \cdots, \phi_{x_n}) d\mathbf{x} \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$, $\phi = \phi(x_1, x_2, \cdots, x_n)$, $\phi_{x_i} = \frac{\partial \phi}{\partial x_i}$ and Ω is a region in n-dimensional space.

The corresponding Euler-Lagrange equation has the following form:

$$\frac{\partial \Phi}{\partial \phi} - \Sigma_{i=1}^{n} \frac{\partial}{\partial x_i} \frac{\partial \Phi}{\partial \phi_{x_i}} = 0 \tag{2.2}$$

The Euler-Lagrange equation has different forms based on the numbers of the involved functions and variables, such as, the Euler-Lagrange equation of single function of single variable and the Euler-Lagrange equation of several function of several variables. The Euler-Lagrange equation can be used according to the following theorem:

Theorem 2.3.3 *If Functional (2.1) attains its extreme when* $\phi = \phi_0(\mathbf{x})$*, then* $\phi = \phi_0(\mathbf{x})$ *must satisfy Equation (2.2).*

A derivation of the Euler-Lagrange equation of single function of single variable is given in Appendix A.

2.3.2.2 3-D Scene Reconstruction Based on Variational Principles

A set of images can be thought of as defining a set of constraints on a 3D scene geometry and radiance or reflection. To reconstruct this 3D scene, a valid 3D scene model that is projected using the camera matrices associated with the input images should produce synthetic images that are the same as the corresponding

real input images. With this variational principle and its variants, many 3D scene reconstruction techniques have been proposed [38, 49, 52, 60, 89, 93, 109, 136, 154, 187]. Some formulations only use texture to infer the 3D shape like [52], some methods combine the informations of texture and contours to reconstruct the 3D geometry such as [38, 49] and some methods formulate the variational frameworks based on the segmentation of the 2D images or the 3D volumes like [93, 154].

Faugeras and Keriven [52] proposed four variational formulations with increasing complexity of the object model and the matching functional for 3D shape reconstruction. The final one which was used to compute the results displayed in this paper is given in the following form:

$$\Psi_4(\mathbf{S},\mathbf{N}) = \int\int \Phi_4(\mathbf{S},\mathbf{N},v,w)d\sigma, \qquad \text{where} \qquad \Phi_4 = -\Sigma_{i,j=1,i\neq j}^{n} \frac{\langle I_i, I_j\rangle}{|I_i|\cdot|I_j|}$$
$$\langle I_i, I_j\rangle = \frac{1}{4pq}\int_{-p}^{+p}\int_{-q}^{+q} [I_i(m_i+m) - \bar{I}_i(m_i)]\cdot\left\{I_j[H(m_j+m)] - \bar{I}_j(m_j)\right\} dm$$
$$+\frac{1}{4pq}\int_{-p}^{+p}\int_{-q}^{+q} \left\{I_i[H^{-1}(m_i+m')] - \bar{I}_i(m_i)\right\}\cdot[I_j(m_i+m') - \bar{I}_j(m_j)]dm',$$
$$|I_i|^2 = \langle I_i, I_i\rangle, \qquad d\sigma = |\mathbf{S}_v \times \mathbf{S}_w| dvdw. \tag{2.3}$$

Here, $\mathbf{S}$ is the object surface embedded in a 3D volume and parameterized by v and w. $\mathbf{N}$ is the inner unit normal to the surface. H is the homography introduced by the tangent plane at a point on the surface. It can be seen that only texture is included in this formulation.

The energy formulation given by Esteban and Schmitt in [49] has the following form:

$$\mathcal{E}(\mathbf{S}) = \mathcal{E}_{tex}(\mathbf{S}) + \mathcal{E}_{sil}(\mathbf{S}) + \mathcal{E}_{int}(\mathbf{S}) \tag{2.4}$$

Here, $\mathcal{E}_{tex}(\mathbf{S})$ includes the information from the texture, $\mathcal{E}_{sil}(\mathbf{S})$ contains the information from silhouettes and $\mathcal{E}_{int}(\mathbf{S})$ is a smoothing term. In this formulation, the information from texture and contours are all taken into account.

The variational formulation allows to combine a data fidelity criterion on the unknown surface with desired regularization term, thus achieving a considerable increase in robustness to image noise. But this combination is also a two edged sword because it is difficult to adjust the regularization behavior [154]. In addition, for many variational formulations, the empty set is always a global minimum and the evolutions are easily trapped on local minima, which make the initialization crucial. Kolev et al. [93] proposed a continuous global optimization framework. This formulation not only contains the usual data fidelity and regularization terms but also includes a convex penalizer which makes it a convex formulation. The energy model proposed in [93] has the following form:

$$E(\phi) = \int_{\mathbf{V}} (\rho_{bck}(\mathbf{x}) - \rho_{obj}(\mathbf{x}))\phi(\mathbf{x}) + \nu\rho(\mathbf{x})|\nabla\phi| + \alpha\theta(\phi(\mathbf{x}))d\mathbf{x} \qquad (2.5)$$

where $\mathbf{V}$ is the volume containing the scene to be reconstructed. $\mathbf{V}$ is divided into two parts $\Omega^{\mathbf{S}}_{obj}$ and $\Omega^{\mathbf{S}}_{bck}$ denoting the interior and the exterior with respect to a surface $\mathbf{S}$. $\phi : \mathbf{V} \rightarrow \{0, 1\}$ is the characteristic function which represent the surfaces implicitly. $\rho(\mathbf{x}) \in [0, 1]$ denotes the probability of a voxel lying on the surface. $\rho_{obj}(\mathbf{x})$, $\rho_{bck}(\mathbf{x}) \in [0, 1]$ describe the probabilities of a point $\mathbf{x}$ belonging to $\Omega^{\mathbf{S}}_{obj}$ and $\Omega^{\mathbf{S}}_{bck}$. $\theta(\phi) = max\{0, 2|\phi - \frac{1}{2}| - 1\}$ is a convex penalizer. For a sufficiently large α, it can enforce ϕ to stay in the interval $[0, 1]$.

Although this formulation is convex, it is based on the assumption that the visibilities to each view have been obtained. On the other hand, the visibilities are approximated based on the visual hull which limits its application.

2.4 Reconstruction Based on Merging Depth Maps

Usually a range image generated from an active vision technique or the depth maps of a view computed in a passive vision technique can not cover the full surfaces of the objects in a scene. The integration of multiple range images or depth maps of multiple views is necessary to obtain a full reconstruction. Many methods have been proposed on surface fusion for the data produced by range finders and depth maps merging for the depth maps computed from the images generated by passive sensors [40, 81, 156, 174, 183].

2.4.1 Fusion of Surfaces Produced by Range Finders

Surface fusion has received considerable attention in the literature for data produced by range finders, where the noise level and the fraction of outliers is typically lower than what is encountered by using passive sensors.

Turk and Levoy [174] proposed a method for combining a collection of range images into a single polygonal mesh that completely describes an object to the extent that it is visible from the outside. They align the meshes and zip together adjacent ones to form a continuous surface that correctly captures the topology of the object and then generate a consensus surface geometry by computing local weighted averages of surface positions on all meshes. Soucy and Laurendeau [156] introduced a similar algorithm in which the integrated surface model is piecewise estimated by a set of triangulations modeling each canonical subset of the Venn diagram of the set of range views. The local models are connected by constrained Delaunay triangulations and then yielding a non-redundant surface triangulation describing all surface elements sampled by the set of range views. A different approach was proposed by Curless and Levoy [40] who employ a volumetric representation of the space and compute signed distance functions from the depth

estimates. The signed distance functions are cumulated into a single one which represents the surface implicitly. Hilton et al. [81] presented a reliable range image integration algorithm based on an implicit surface representation which explicitly takes into account boundaries and holes.

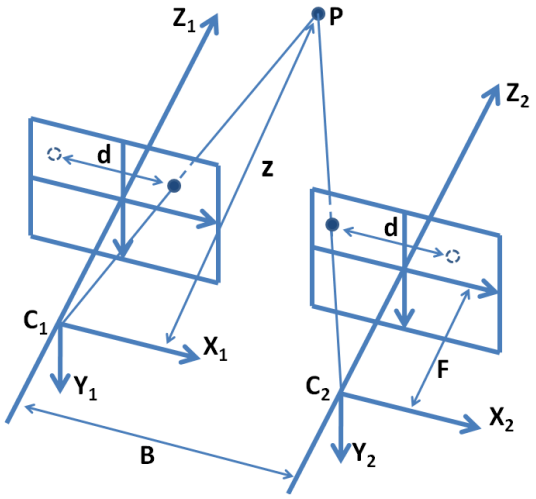


Figure 2.7: The camera configuration in a conventional stereo rig.

2.4.2 Stereo Reconstruction

Stereo reconstruction is one of the most common techniques for inferring scene geometry from two images with different viewpoints. It dates back to the 1970's when Marr and Poggio [114] attempted to solve the stereo problem based on human-stereopsis. In a conventional binocular stereo rig, the two cameras have the same intrinsic parameters and the two optical axes are parallel to each other and the two image planes lie in the same plane (see Figure 2.7). In this camera configuration, if a 3D point P can be seen by both cameras, the images of this point will be registered in the same row and the only difference between the locations of the pixels is in the horizontal location. The relationship between a pixel (x,y) in reference image and the corresponding pixel (x', y') in the matching

image is then given by

$$x' = x + d(x, y), \quad y' = y \tag{2.6}$$

where $d(x, y)$ is the disparity at pixel (x, y), which is defined as the displacement of the corresponding pixels in the two images.

If the two cameras used in the stereo vision have a more general configuration, the disparities are found along the epipolar lines. In practice, in many stereo reconstruction methods, the images are preprocessed with some image rectification algorithms [9, 63] before building the correspondences between these two images. As a result, the searching for building the correspondence between images is still performed along a horizontal line.

Given B the length of the stereo baseline, F the focal length of the cameras and d the disparity between the images of the point, the conventional stereo vision determines the depth z of a 3D point P with the following formula:

$$z = \frac{BF}{d} \tag{2.7}$$

In fact, the central problem in stereo reconstruction is the stereo correspondence. A lot of work has been devoted here. Early algorithms on stereo vision match sparse image features especially edge features [8, 10, 114, 130]. Edge matching is useful because the edges detected in an image describe important geometry of the scene. But if the correspondences are only built between the sparse image features, a problem arising is that how to fill the gaps between the features? Some assumptions such as continuity can be used to do it, but these assumptions are only valid in specific situations. In contrast to feature matching in stereo vision, the dense matching methods try to find a correspondence between every pixel in the stereo images [16, 17, 51, 88]. Kanade and Okutomi [88] proposed a stereo matching algorithm with an adaptive window. An appropriate window is selected by evaluating the local variation of the intensity and disparity

to guarantee including enough intensity variation for matching and to avoid the effects of projective distortion. Window-base methods are usually called local methods because they attempt to optimize each point separately by aggregating the cost within the window. On the other hand global techniques in stereo vision [17, 18, 26, 57, 141, 141] typically compute disparity values for every point and then minimize an energy function based on the costs for every point along with a smoothness term.

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{2.8}$$

where $E_{data}(d)$ measures how well the disparity function d agrees with the input image pair and $E_{smooth}(d)$ encodes the smoothness assumptions made by the algorithm.

An exhaustive comparison between different stereo algorithms and experiment evaluations is given in [142].

2.4.3 Fusion of Depth Maps Computed from Images

Among the first approaches to fuse depth maps computed from passive data was that of Fua [58] where he adopted a particle based representation. The positions and orientations of the particles are initialized from the depth estimates and then refined by minimizing an image-based objective function. Spurious particles are eliminated and those that appear to belong to the same global surfaces are clustered. Koch et al. [91] proposed an automatic 3D surface modeling system that extracts dense metric 3D surface from an uncalibrated video sequence. First the dense correspondence maps are computed between adjacent image pairs and then all possible image point correspondences over the image sequence are linked by a linking algorithm. The position of the point is updated using the wider baseline, reducing the sensitivity to noise. Koch et al. [92] also presented a volumetric approach for fusion. Given depth maps for all images, the depth estimates for all pixels are projected in the voxelized 3D space. Each depth estimate votes for a

voxel probabilistically and the surfaces are extracted by thresholding. Hernandez et al. [79] compute the probability that a point in a volumetric grid is visible from depth maps and segment the volume into foreground and background using graph cuts. Goesele et al. [65] presented a two-stage approach which computes the depth maps for each reference view and then merges depth maps into a single one to extract the surface. In the first step, normalized cross-correlation is computed for each depth estimate between the reference view and several target views. The depth estimates are rejected if the cross-correlation is not large enough for at least two target views. The remaining depth estimates are used for surface reconstruction using the technique of [40]. Because of the noise of the input images, the depth estimates are frequently rejected in the first step leading to many holes in the final reconstruction.

2.5 Image Features and Region Growing Methods

Image matching is a fundamental aspect of many problems in computer vision, including 3D model reconstruction, object recognition, motion tracking, image alignment, camera self-calibration and so on. Among all kinds of image matching methods, such as histogram-based image matching, region-based image matching, feature point matching is the most popular one. A lot of work has been devoted to feature detecting and matching [113, 123, 144, 193]. But feature matching techniques usually yield only a sparse set of matching features, which could be sufficient for applications like camera self-calibration or object recognition. But for reconstructing a dense 3D model of a scene, they are usually not enough. Some 3D reconstruction techniques use matching features as a constraint [158] and some use them to initialize the shape and obtain the final reconstruction by combining other algorithms [99, 161]. There is a group of methods which regard the matching

features as seed points(regions) and repeatedly expand them to nearby pixel or surfels(surface elements) to reconstruct a dense surface. This group of methods is usually called "region growing methods" which are summarized in this section.

2.5.1 Image Features

A local feature of an image is an image pattern which differs from its immediate neighborhood. Image features usually include the following four different types: the edges, the corners(interest points), the blobs(region of interest points) and the ridges.

There is not a universal definition of what constitutes an image feature and the exact definition often depends on the problem, the methods used for detecting the features or the type of applications. For example, if a search-based method is used to detect edges in an image, then the edges can be defined as the pixel with their gradient magnitude larger than a threshold. On the other hand, if a zero-crossing based method is used to detect edges, then the edges can be defined as the pixels where zero-crossings happen in a second-order derivative expression.

Many methods have been proposed to detect the image features which can be divided into a few categories, contour curvature based methods, model-based methods, multi-scale methods, intensity based methods and so on. A very good survey is given in [175]. Here we just mention a few important ones. Beaudet [13] proposed a Hassian-based detector which explores the Hessian matrix of the intensity surface. The determinant of this matrix reaches a maximum for blob-like structures in the image. This method has been extended by Dreschler and Nagel in [45] and Zuniga and Haralick in [197], where the feature points are located at the zeros crossing of a curve joining local extrema of the Hessian determinant around a corner. Similar ideas were proposed in [50, 149]. The Harris detector [75] has been a popular interest point detector for decades. The Harris detector computes the locally averaged moment matrix obtained from the image

gradients and then combines the eigenvalues of the moment matrix to compute a corner "strength", of which the maximum values indicate the corner positions. Compared with the Moravec detector [122], the Harris detector is a more desirable one in terms of detection and repeatability rate. Despite the high computational cost, the algorithm is widely used in practice. Harris detector has been extended in numerous papers [143, 195]. A new popular interest point detector called the Scale Invariant Feature Transform (SIFT) was proposed by Lowe [112]. In this algorithm, a scale-space pyramid was built using the Difference-of-Gaussian function. The local 3D extrema in the pyramid representation determine the localization and the scale of the interest points.

2.5.2 Region Growing Methods

There are several methods in the literature which reconstruct the 3D models from 2D images based on region growing [61, 73, 110, 133, 194]. Compared with others, the algorithms proposed in [61] and [73] achieve much better reconstruction. Here I just give a brief explanation and comparison of the two more recent ones [61] and [73].

In the algorithm proposed in [73], first a set of surface elements (surfels) are computed in the form of planar disks with the technique proposed in [72]. In this technique, scene planes are denoted by the four parameters $\mathbf{N}^T=(n_0, n_1, n_2, d)$ with $\mathbf{N}^T\mathbf{P}=0$ for all scene points $\mathbf{P}$ lying on $\mathbf{N}$. Here $\mathbf{P}$ is represented in homogeneous coordinates. For a set of pixels in Ω in the reference image, the corresponding pixels in any comparison image are calculated by a homography introduced by $\mathbf{N}$. Start from an initial estimate of the plane parameters $\tilde{\mathbf{N}}$, the correct plane parameters $\mathbf{N}$ are computed by a fitting algorithm. Concretely they are computed by iteratively minimizing a sum of square differences (SSD) objective function of the intensities of the pixels in Ω and the associated ones in the comparison images. After the seed disks are generated, they are used to

initiate the surface reconstruction process on any not yet recovered part of the scene. The disks that are used to approximate the unknown surface of a scene can be divided into two groups: active or inactive. A disk is called inactive either if its geodesic neighborhood on the surface is completely covered with disks or if it is not possible to smoothly expand the recovered region at this particular position. Otherwise a disk is called active. The main idea of the surface growing approach is to add new disks to complete the neighborhood of active disks followed by a correcting step of the plane parameters using the fitting algorithm mentioned above, and then the recovered region is iteratively expanded by growing further disks in tangential direction until a disk rotates by more than a threshold during the fitting step.

The region growing algorithm proposed in [61] has a reconstruction procedure of 3 steps: matching, expansion and filtering. In the matching step, first, the corners and blob features are detected in each image using Harris and Difference-of-Gaussian operators and then they are matched across multiple images to reconstruct a sparse set of patches. In the second step, new neighbors are added to existing patches until they cover the surface visible in the corresponding view. In the last step, two filtering procedures are applied to the reconstructed patches to further enforce visibility consistency and remove erroneous matches.

There are two main differences between these two algorithms. First, in the algorithm proposed in [61] the initial patches (which are called seed disks in [73]) are generated based on the matching of features which are detected by Harris and Difference-of-Gaussian operators. Unlike that in [73] where the fitting algorithm is based on the matching of an unconstrained homography H with 8 degrees of freedom using the technique in [11]. There are no features detected and exploited in [73]. The second difference is the including of the filtering step and the repetition of the second and third steps in the procedure in [61]. In [73], there is no guarantee that the plane fitting does not get stuck in a local minimum and the seed disk hence does not lie on the scene surface. Further mechanisms have

to be employed to detect such cases. In [61], the initial patches are built upon the matching of features, the initialization of the patch parameters can be closer to the corrected ones than that of the seed disks in [73]. So the convergence of the fitting algorithm in [61] can work better. What is more, because of the including of the filtering step and repetitively using of the filtering step, the algorithm proposed in [61] can tolerate more erroneous patches generated in the first step and obtain better results in terms of accuracy and completeness than those in [73].

2.6 Conclusion

In this chapter, a review on the methods for dense shape reconstruction by multi-view stereo is given. They are divided into four classes: voxel-based 3D volumetric approaches, partial differential equations (PDEs) based surface evolution methods, the algorithms that compute and merge depth maps and the techniques to reconstruct the 3D surfaces by region growing. Our work of solving multiview stereo with a large baseline can be seen as a combination of the first three classes. In our work we employ the volumetric data representation. Because we use level sets to represent the surface, with volumetric data representation it is easier to embed the level set into a four-dimensional space. In our work, the depth maps for each pixel is generated by photoconsistency measuring as done in the Space Carving algorithm. Then the visibility functions are built based on the depth maps. Although we do not do the depth maps merging in our work, we need to integrate the visibility functions of each central view into a global visibility function with a scheme of visibility voting. This step is different to the depth maps merging but they fulfill the same task of integrating the part surface reconstructions into a full surface reconstruction. In our work, the surface reconstruction problem is formulated as a energy minimization problem. Starting from a initialization, the final surface is obtained by evolving the initial surface by some

partial differential equations. In this step, we mimic the ideas of the second class into our work. The only class of methods which are not connected very close to our work are the methods based on image growing. But there are at least two reason to include them here: first, feature extraction and matching are important topics in computer vision and many efforts have been devoted on them. The corresponding algorithms have been used extensively. Second, we include this class of methods here for the completeness of the review.

All the methods reviewed in this chapter achieve 3D shape reconstruction based on the information of 2D images. To understand how a 2D image is captured by a imaging device is important to fulfill the reconstruction task. The image formation procedure will be presented in the next chapter.

Chapter 3

Image Formation

3.1 Introduction

Computer vision is the science and technology of machines that extract information from images for recovering a description of the scene in space. It is, in some ways, solving the inverse problem of image formation which studies how a scene gives rise to images. Therefore, it is necessary to get familiar with the image formation process before designing computer vision algorithms.

There are two parts of the image formation process. The first part is the geometry of the image formation which determines where in the image plane the projection of a point in the scene will be located. The second part is the physics of light which determines the brightness of a point in the image plane based on the illumination condition, surface properties and the characteristics of the imaging system.

To give a clear explanation of the image formation process, we start this chapter with the light trajectory in image formation process and which is then followed by the pinhole camera model and the finite aperture camera models. After this, the image formation procedure is described from an algebraic viewpoint by giving the projection matrix in homogeneous coordinates. This chapter will

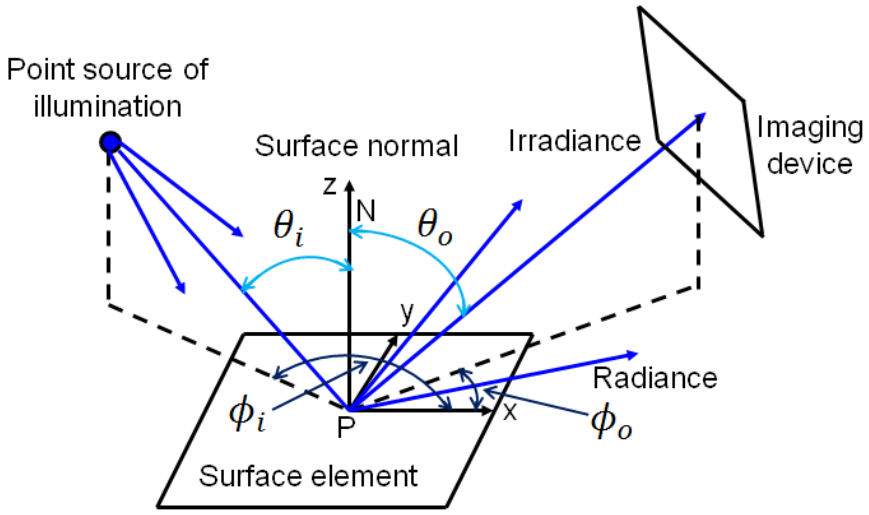


Figure 3.1: The light trajectory in the image formation process. In this figure, **P** is a point on the surface and $\overrightarrow{\mathbf{PN}}$ is the normal vector of the surface at point **P**. A ray starts from an illumination source and hits the scene surface at point **P** with incident angle θ_i. This ray will be reflected along different directions and some of them will reach the imaging device. The irradiance measured by the imaging device along certain direction can be computed with the corresponding *bi-directional reflectance distribution function*.

be terminated by a brief explanation of camera calibration.

3.2 The light trajectory in Image Formation

As is mentioned in Section 3.1, the physics of light is part of the image formation process. It is necessary to understand how the light emitted from an illumination source is reflected by an object surface and how it is measured by an imaging device. This procedure is illustrated in Figure 3.1.

In Figure 3.1, a point light source is used. In general, the geometric shape of a light can also be a line or a patch with finite area. An orthogonal coordinate system in 3D is attached to the surface point **P**, which has the z-axis and xy-plane coinciding with the surface normal $\overrightarrow{\mathbf{PN}}$ and the tangent plane of the surface at point **P** respectively. The direction of a light ray can be represented by two angles, the angle between the light ray and the surface normal and the angle

between the projection of the light ray on the xy-plane and the x-axis, such as (θ_i, ϕ_i) and (θ_o, ϕ_o) in Figure 3.1.

The amount of light can be measured with *radiance*. Here we adopt the definition of *radiance* given in [148].

Definition *(Radiance). Radiance measures the amount of energy traveling at some point in a specified direction, per unit time, per unit area perpendicular to the direction of travel, per unit solid angle.*

In the above definition, the notion of *solid angle* can be defined as

Definition *(Solid angle). The solid angle of a cone of a certain direction is the area cut out by the cone on the unit sphere.*

For most materials in practice, the proportion of a coming light reflected by their surface in a certain direction can be expressed by the *bi-directional reflectance distribution function* (BRDF) $\beta_P(\theta_i, \phi_i, \theta_o, \phi_o)$ [77, 115, 126, 140, 180]. This is a function of two directions, the direction of the incident ray and the the direction of the outgoing ray. For an outgoing light direction (θ_o, ϕ_o), the radiance at the point $\mathbf{P}$ is given by the integration of BRDF against all incident directions (θ_i, ϕ_i)

$$\rho_P(\theta_o, \phi_o) = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \beta_P(\theta_i, \phi_i, \theta_o, \phi_o) L_P(\theta_i, \phi_i) cos\theta_i sin\theta_i d\theta_i d\phi_i \tag{3.1}$$

where $L_P(\theta_i, \phi_i)$ is the radiance of the incident light ray at point P coming from direction (θ_i, ϕ_i).

There are certain materials whose BRDFs do not depend on the outgoing direction, i.e., the BRDF is constant between any incident and outgoing direction pair at each surface point. The surfaces of this kind of materials are called *Lambertian surface*. A Lambertian surface is one of the fundamental assumptions for most well-studied vision problems. Its usage will be seen in Chapter 6. For

a Lambertian surface point **P** with constant BRDF β_P the radiance along any direction is the same as

$$\rho_P = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \beta_P L_P(\theta_i, \phi_i) cos\theta_i sin\theta_i d\theta_i d\phi_i \tag{3.2}$$

The last stop of the light trajectory in image formation process is an imaging device, which measures how much light is received by its sensor. In radiometry, the term *irradiance* is used to describe the amount of light incident on a surface. It is defined in [148] as

Definition *(Irradiance). Irradiance measures incident light power per unit area.*

A typical imaging device is the camera. Some camera models will be described in the following sections.

3.3 Pinhole Camera Model

The pinhole camera is the simplest geometric model of a device to form an image of a 3D scene on a 2D surface with a single small aperture, effectively a box with a small hole on one side. Light from a scene passes through this single point and projects an inverted image on the opposite side of the box (Figure 3.2).

3.3.1 Pinhole Camera Geometry

From the schematic of the pinhole optical system illustrated in Figure 3.2, it can be seen that all the optical rays falling onto the image plane are passing through the small aperture. Since the aperture is assumed very small in the ideal pinhole camera model, it can be described by a point **C**, which is called the *projection center* or the *optical center* of the camera. Lights go along straight lines, so the geometric relationship between a 3D point **P** and its projection **p** on the image plane can be derived. To describe this relationship, a reference frame for a camera,

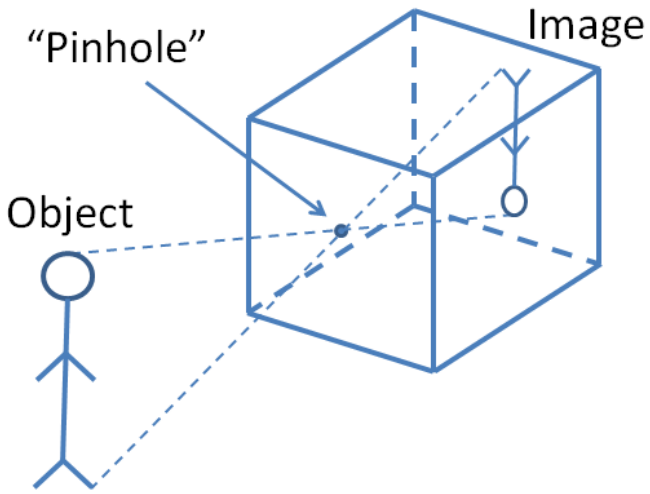


Figure 3.2: A schematic of the pinhole camera. Light rays from an object pass through a small hole to form an image.

called *camera reference frame*, should be set up. A camera reference frame is an orthogonal coordinate frame attached to the optical center with its z-axis being the optical axis (the axis passing through the optical center and perpendicular to the image plane) and x- and y-axes parallel to the image row and column. The intersection of the optical axis with the image plane is called *image center* (Figure 3.3). The distance from the optical center to the image plane is called the focal length of the pinhole camera. If the focal length of a pinhole camera is f then, in the camera reference frame, the image plane coincides with the plane $z = -f$. Therefore, for a 3D point P with coordinates $(X, Y, Z)^T$ in the camera reference frame and its projection p with coordinates $(x, y, -f)^T$, the coordinates' relationship between them is (see left image in Figure 3.3)

$$\begin{aligned} x &= -f\frac{X}{Z} \\ y &= -f\frac{Y}{Z} \end{aligned} \tag{3.3}$$

The negative signs can be get rid of by simply flipping the image coordinates $(x, y) \to (-x, -y)$. This is equivalent to placing the image plane in front of the

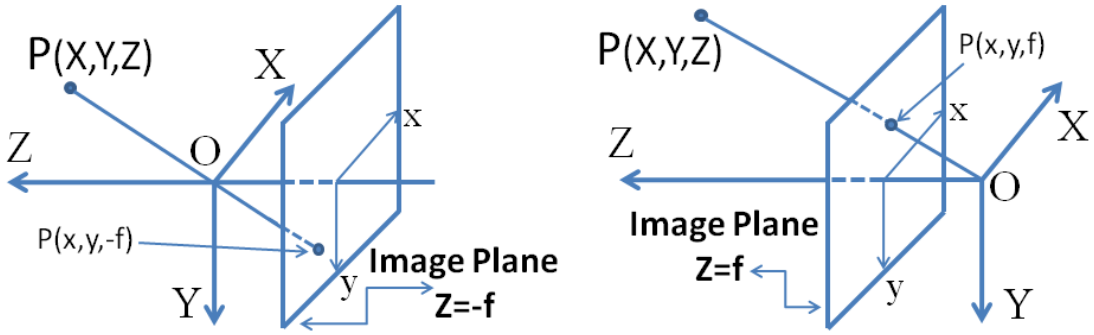


Figure 3.3: The geometry of the ideal pinhole camera model. It describes the relationship between a 3D point $(X, Y, Z)^T$ and its corresponding 2D projection $(x, y)^T$ onto the image plane (left) and the flipped image plane (right).

optical center at $z = f$ (see the right image in Figure 3.3). Now the relationship between (X, Y, Z) and (x, y) is

$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z} \tag{3.4}$$

The pinhole camera model given in this section is an ideal image formation model. Up to a certain degree, the smaller is the aperture, the sharper is the image, but the image is dimmer. Usually forming an image with a pinhole camera needs a long exposure time up to hours and even days. But an extremely small aperture can produce significant diffraction effects due to the wave properties of light, resulting in a blurred image. Additionally, vignetting, which is the reduction of an image's brightness at the periphery compared to the image center, occurs as the diameter of the aperture is down to the thickness of the material in which it is punched. This is because the sides of the aperture obstruct the light entering at any direction other than the perpendicular ones. For a detailed mathematical treatment, the readers can refer to [172].

In practice, the aperture must be larger to admit more light. But, on the other hand, with a wide pinhole, light from a point on the scene surface spreads across

the image, also leading to blurred images. So to duplicate the pinhole geometry without experiencing the undesirable aspects of the apertures, a lens or a group of lenses are placed in the aperture to focus the bundle of rays from each scene point onto the corresponding point in the image plane. This leads to the camera models with finite apertures, which will be discussed in the next section.

3.4 Camera Models with Finite Aperture

A pinhole camera would be excellent for imaging except for a few serious limitations as pointed out in the later part of the last section. One of the limitations is the image resolution. While the geometric optics states that making the pinhole smaller improves the resolution of the image, this also reduces the amount of incoming light. In addition, diffraction limits the effectiveness of shrinking the hole. How to modify a pinhole camera model to admit more light and at the same time give higher resolution? The answer exists in putting a simple convex lens at the aperture with the focal length equal to the distance to the image plane. This allows the hole to be enlarged and at the same time keep the geometry almost the same as that of a ideal pinhole camera model (Figure 3.4).

In practice, usually a compound lens, which includes a number of optical lens elements, is used in a camera. The additional elements allow the lens to reduce various optical aberrations, but the principle of operation remains the same: a pencil of rays is collected at the entrance pupil and focused down from the exit pupil onto the image plane. So in this section, a thin lens is used to describe the image formation process in camera models with finite apertures.

3.4.1 The Thin Lens

As depicted in Figure 3.4, a camera consists of an optical lens and a image plane which contains an array of sensor elements that measure the amount of light in-

cident on a particular location on the image plane. In optics, a lens is a device that alters light propagation direction via diffraction. A thin lens is a lens with a thickness that is negligible compared to its focal length (i.e., approximately planar). The thin lens assumption can simplify ray tracing calculations by ignoring the optical effects coming with the thickness of lenses. The line which coincides with the axis of rotational symmetry of the lens and is perpendicular to the image plane is called the *optical axis* of the camera. The intersection of the optical axis with the plane of the lens is called *optical center*. There are a few parameters in Figure 3.4 that should be specified. f is the focal length of the lens, which depends on the shape and material of the lens. u is the distance from a point on the object surface to the lens, called *object distance*. The light rays emanated from a surface point **P** will converge to a point **Q** after being diffracted by the lens. The distance from **Q** to the lens is called *image distance* represented by v. The distance from the image plane to the lens, called *focus setting*, is represented by v_i which could be different from v. Under the assumption of paraxial rays, the relationship between focal length f, object distance u and image distance v is given by the thin lens equation [21]. It is a mathematical idealization.

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \tag{3.5}$$

For a certain point **P** on the surface of the object, if $v_i = v$, its image is still a point. But, if $v_i \neq v$, the image of point **P** will be spread out. Assuming a circular lens, the energy coming from a point that is not in focus will distributed within a disc, called the *circle of confusion* (COC). By similarity of triangles, the radius of the COC can be computed as

$$b_1 = b_2 = \frac{D}{2}\left|1 - \frac{v_i}{v}\right| \tag{3.6}$$

where D is the aperture of the lens. b_1 and b_2 are the distances as depicted in

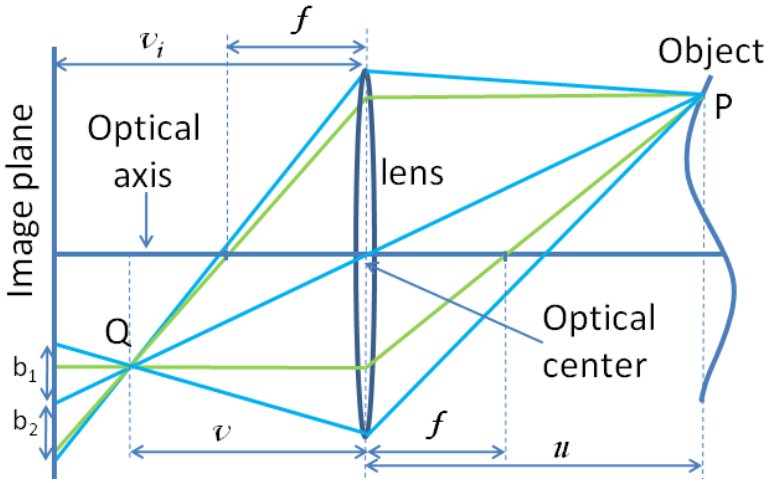


Figure 3.4: The schematic of the camera models with finite apertures. In this figure, f is the *focal length* of the lens, u is the *object distance*, v is the *image distance* and v_i is the distance from the image plane to the lens. If the lens is a circular one, when $v_i \neq v$ the energy from a point will distribute within a circle and then $b_1 = b_2$ is the radius of the circle.

Figure 3.4. The absolute value in this equation includes the cases where $v < v_i$.

3.4.2 Imaging Models for Cameras with Finite Apertures

In Section 3.4.1, it has been pointed out that when the *focus setting* v_i does not satisfy the thin lens Equation (3.5) the image of a surface point is no longer a point but a defocus pattern such as a disc or what it is called a circle of confusion. The corresponding region of the defocus pattern on the image plane is called the *confusion region*. Then the energy radiated from a surface point will be distributed within the confusion region. The function h, which encodes the energy distribution within the corresponding confusion region is called *point spread function*. Obviously, h depends on the image point $\mathbf{p}$, surface point $\mathbf{P} = [\mathbf{x}^T, u(\mathbf{x})]^T$ with $\mathbf{x}$ as a parameterization of the surface. In addition, h also depends on the focus setting v_i. So it is often written as $h^{v_i}(\mathbf{p}, \mathbf{x})$ to indicate its dependency on $\mathbf{p}$, $\mathbf{x}$ and v_i. Suppose the point source at $\mathbf{x}$ emits light with infinitesimal intensity $r(\mathbf{x})d\mathbf{x}$, then the energy given by the surface point $[\mathbf{x}^T, u(\mathbf{x})]^T$ to the image point

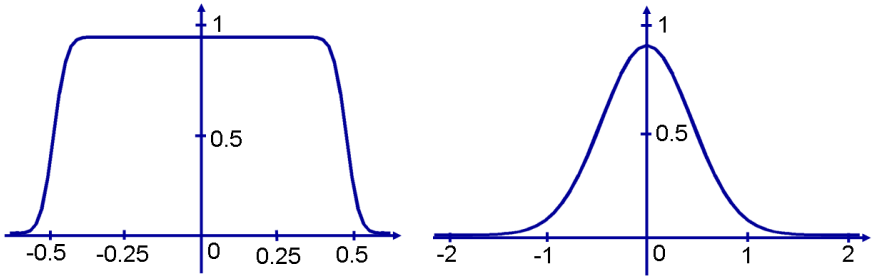


Figure 3.5: Two examples for point spread function. A smoothed vision of a pillbox function in 1D (left). A Gaussian function in 1D (right).

$\mathbf{p}$ is $h^{v_i}(\mathbf{p}, \mathbf{x})r(\mathbf{x})d\mathbf{x}$. The image is formed as the combination of the contributions from each point source. It is common to assume that sources combine linearly, so the image is simply the sum of the contribution from each source. So the image model can be written as

$$I(\mathbf{p}) = \int_{\mathbb{R}^2} h^{v_i}(\mathbf{p}, \mathbf{x})r(\mathbf{x})d\mathbf{x} \qquad \mathbf{p} \in \Omega \subset \mathbb{R}^2 \tag{3.7}$$

where function $r : \mathbb{R}^2 \to [0, \infty)$ indicates the energy radiated from the corresponding surface point.

If we assume that the energy coming from a surface point is uniformly distributed within the corresponding confusion region, then the point spread function is a pillbox function, which is the indicator function of the confusion region normalized by its area. Usually this assumption is valid for a large aperture camera. For a relatively small aperture camera, the point spread function is usually set as a Gaussian. Based on our experimental results, for a small aperture camera, Gaussian is a better choice than pillbox. Because within the confusion region of the image of a light ray, we can see the center part is obviously brighter than the boundary. But choosing the pillbox function as the point spread function can simplify the computation, so when the aperture is large enough and the brightness attenuation from the center to the boundary within a confusion region is not

significant, a pillbox function is often used. For the stability in numerical scheme, the pillbox function is often replaced with a regularized version. A regularized version of a pillbox function in 1D is depicted with the left image in Figure 3.5, while the right image in this figure shows an example of Gaussian in 1D.

3.5 Camera Projection Matrix

3.5.1 Homogeneous Coordinates

Homogeneous coordinates are an analytical tool most suitable for tackling problems in projective geometry, playing a role equivalent to the one Cartesian coordinates play to Euclidean geometry [118]. Given a point X in the n-dimensional space with Cartesian coordinates $(X_1, X_2, \ldots, X_n) \in \mathbb{R}^n$, the corresponding homogeneous coordinates of X are the set of $(n+1)$-tuples $\{w(X_1, X_2, \ldots, X_n, 1), \forall w \in \mathbb{R}\backslash\{0\}\}$. Conversely, given the homogeneous coordinates of n-dimensional point X as $(X_1, X_2, \ldots, X_n, X_{n+1}) \in \mathbb{R}^{n+1}\backslash\{0, 0, \ldots, 0)\}$, the Cartesian coordinates of X $(X_1, X_2, \ldots, X_n)/X_{n+1}$, if $X_{n+1} \neq 0$. If $X_{n+1} = 0$, the point X is said to be at infinity in direction $(X_1, X_2, \ldots, X_n)$, and it cannot be represented in Cartesian coordinates.

3.5.2 The Calibration Matrix of a Perspective Camera

If the world and image points are represented by homogeneous coordinates, then the pinhole camera model can be easily expressed as linear mapping between their homogeneous coordinates. In particular, Equation (3.4) can be written in

homogeneous coordinates in terms of matrix multiplication as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.8}$$

The above expression can be written in a more compact form as

$$\mathbf{x} = \mathsf{P}\mathbf{X} \tag{3.9}$$

where $\mathbf{x}$ is a 3-vector representing the homogeneous coordinates of an image point, $\mathbf{X}$ is a 4-vector representing the homogeneous coordinates of the corresponding world point in the camera coordinate system and the 3×4 matrix P is the *camera projection matrix*.

In (3.8), it is assumed that the origin of coordinates in the image plane is at the principal point (the intersection of the optical axis and the image plane). In practice, it is not. See Figure 3.6. Suppose the coordinates of the image center in the image coordinate system are (u_0, v_0), then the mapping of a world point in the camera coordinate system to a image point in the image coordinate system is

$$(X, Y, Z)^T \rightarrow (fX/Z + u_0, fY/Z + v_0)^T \tag{3.10}$$

The above expression can be written conveniently in homogeneous coordinates as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX + Zu_0 \\ fY + Zv_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.11}$$

The pinhole camera model expressed by Equation (3.11) assumes that the

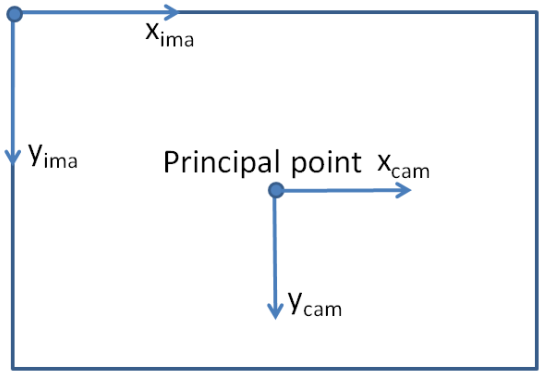


Figure 3.6: Image coordinate system (x_{ima}, y_{ima}) and camera coordinate system (x_{cam}, y_{cam}).

image coordinates are Euclidean coordinates having equal scales in both axial directions. In the case of CCD cameras, the image coordinates can be measured in pixels. Since a pixel is not necessarily a square, there is the extra effect of introducing unequal scale factors in each direction when image coordinates are measured in pixels. If the number of pixels per unit distance in coordinates are m_x and m_y in the x and y directions, then the mapping of a 3D point from the camera coordinates to pixel coordinates can be written as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} \alpha_x X + Z x_0 \\ \alpha_y Y + Z y_0 \\ Z \end{pmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.12}$$

where $\alpha_x = f m_x$ and $\alpha_y = f m_y$ represent the focal length of the camera in terms of pixel dimensions in the x and y direction respectively. (x_0, y_0) are the coordinates of the image center in the image coordinate system in terms of pixel dimensions.

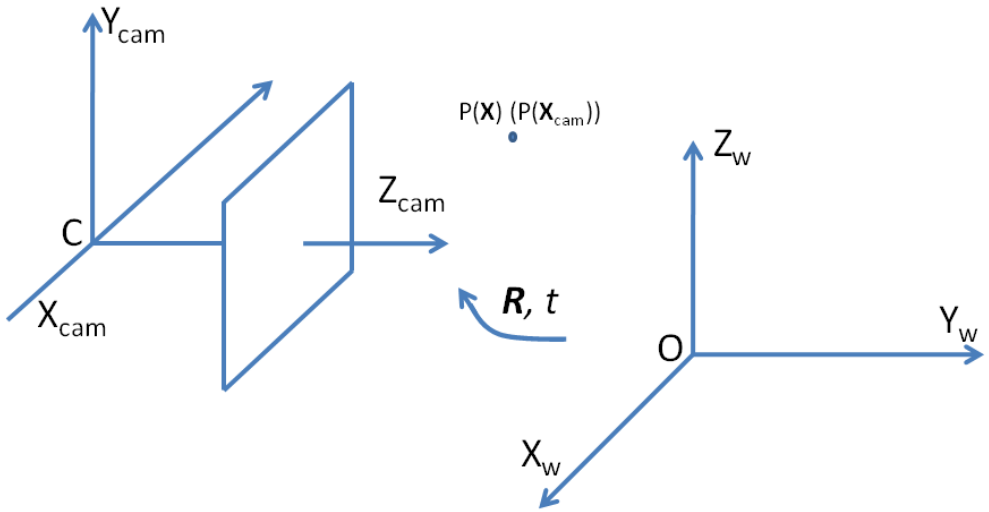


Figure 3.7: The Euclidean transformation between the world and camera coordinate frames.

The matrix

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

is called the *camera calibration matrix*. If the lens skew is taken into account, a skew parameter s should be added into the camera calibration matrix, and the camera calibration matrix can be written as

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Now, Equation (3.12) has the concise form

$$\mathbf{x} = K \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X}_{cam} \tag{3.15}$$

where $\mathbf{X}_{cam}$ are the homogeneous coordinates of a 3D point in a camera coordinate frame.

3.5.3 Camera Rotation and Translation

In the previous section, the mapping is expressed between the *camera coordinate frame* and the *pixel coordinate frame*. In general, especially when taking multiple images of a scene from different view points, points in space are expressed in terms of a common Euclidean coordinate frame, which could be different to any camera coordinate frame. This frame is known as the *world coordinate frame*. The camera and world coordinate frames can be related via a rotation and translation (see Figure 3.7). For a point $\mathbf{P}$ with the inhomogeneous coordinates $\mathbf{X}$ in the world coordinate frame and $\mathbf{X}_{cam}$ in the camera coordinate frame respectively, the relationship between $\mathbf{X}$ and $\mathbf{X}_{cam}$ can be written as $\mathbf{X}_{cam} = R(\mathbf{X} - \mathbf{C})$, where $\mathbf{C}$ represents the coordinates of the camera center in the world coordinate frame and R is the rotation matrix representing the orientation of the camera coordinate frame with respect to the world coordinate frame. In homogeneous coordinates, this equation can be written as

$$\mathbf{X}_{cam} = \begin{bmatrix} R & -R\mathbf{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\mathbf{C} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{3.16}$$

Substituting this into Equation (3.15) leads to the formula

$$\mathbf{x} = KR \begin{bmatrix} \mathbf{I} & -\mathbf{C} \end{bmatrix} \mathbf{X} \tag{3.17}$$

where $\mathbf{X}$ are the homogeneous coordinates of a 3D point in a world coordinate frame. The parameters contained in K are called *intrinsic* parameters of a camera and the parameters contained in R and $\mathbf{C}$ are *extrinsic* parameters of a camera.

Now the *camera projection matrix* is

$$\mathsf{P} = KR \begin{bmatrix} \mathbf{I} & -\mathbf{C} \end{bmatrix} \tag{3.18}$$

To fulfill a computer vision task, for a given view captured by a camera, it is often required that the parameters of the camera are known. This is the problem of camera calibration, which will be discussed briefly in the next section.

3.6 Camera Calibration in Computer Vision

Camera calibration is the process of determining the values of the extrinsic and intrinsic parameters of the camera. It is an essential component for the extraction of precise and reliable 3D metric information from images, especially in 3D reconstruction and recognition, which rely directly on the knowledge of the camera parameters. Many efforts have been devoted on this topic and there is an extensive body of literature on the calibration of digital cameras [3, 36, 173, 184, 192] and there are publicly available softwares for camera calibration such as *Camera Calibration Toolbox for Matlab* developed by Jean-Yves Bouguet [2].

To tackle the problem of camera calibration, a camera model should be chosen first. Unsurprisingly the pinhole camera model will be used in this chapter for describing the process of camera calibration. For an ideal pinhole projection model (without lens distortion), the parameters to be calibrated are the ones contained in the matrices K, R and C included in Equation (3.18). This procedure is usually implemented by computing the projection matrix first and recovering the intrinsic and extrinsic parameters from the projection matrix. The basic equations for calibrating an ideal pinhole projection camera are described in the next section.

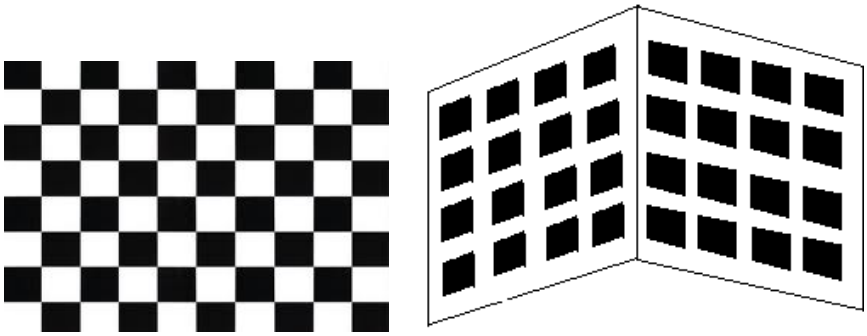


Figure 3.8: Two examples of camera calibration pattern. A planar calibration target with checkerboard patterns (left). Two planes at right angle with Tsai grid (right).

3.6.1 Basic Equations in Camera Calibration

The key idea behind camera calibration is to exploit the known coordinates of a sufficient number of corresponding image and world points to build a system of projection equations and then solve this system to find the camera parameters. In order to know the world points' coordinates, a *calibration pattern* is usually used in the camera calibration process. A calibration pattern is a 2D or 3D object of known geometry and used for generating image features which can be located accurately. Figure 3.8 gives two examples of camera calibration pattern.

With the help of a calibration pattern, it can be assumed that a number of point correspondences between 3D points $\mathbf{P}_i$ with coordinates (X_i, Y_i, Z_i) and their 2D image points $\mathbf{q}_i$ with coordinates (x_i, y_i) are known. For an ideal pinhole camera model, the purpose is to find the camera projection matrix P, given in Equation (3.18), such that

$$\begin{pmatrix} \omega x_i \\ \omega y_i \\ \omega \end{pmatrix} = \mathsf{P} \begin{pmatrix} X_i \\ Y_i \\ Z_I \\ 1 \end{pmatrix}, \qquad \omega \in \mathbb{R}\backslash\{0\}$$

that is

$$x_i = \frac{\omega x_i}{\omega} = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}}$$
$$y_i = \frac{\omega y_i}{\omega} = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} \tag{3.19}$$

The matrix P is defined up to an arbitrary scale factor and has therefore only 11 independent entries, which can be determined via a homogeneous linear system

$$A\mathbf{P} = 0 \tag{3.20}$$

If N matches are given, then we have

$$A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_NX_N & -x_NY_N & -x_NZ_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_NX_N & -y_NY_N & -y_NZ_N & -y_N \end{bmatrix}$$

and

$$P = [p_{11}, p_{12}, \cdots, p_{33}, p_{34}]^T.$$

The number of matching pairs N can be much larger than twelve, and then Equation (3.20) is a overdetermined system of linear equations, which can be solved through *least square* techniques or SVD (short form for *singular value decomposition*).

Knowing the projection matrices of cameras is sufficient for most computer vision tasks. But if necessary, it is easy to recover the individual camera parameters from the projection matrix. See details in [172].

3.6.2 Radial Lens Distortion Modeling

If the radial lens distortion can be neglected, the measured coordinates of the feature points can be used in the corresponding pairs for solving Equation (3.20). But for cheap or short focal length lenses, the observed radial distortions can be significant. For some computer vision applications requiring accurate measurement, they can not be ignored. So there are many camera calibration algorithms that include the lens radial distortion modeling [42, 78, 107, 173, 182]. Radial distortion is typically modeled as

$$\begin{cases} x_c = x_d(1 + k_1 r_d^2 + k_2 r_d^4 + \cdots) \\ y_c = y_d(1 + k_1 r_d^2 + k_2 r_d^4 + \cdots) \end{cases}$$

where (x_d, y_d) are measured coordinates with radial distortion, (x_c, y_c) are corrected coordinates without distortion and $r_d^2 = x_d^2 + y_d^2$. k_i, with $i = 1, 2, \cdots$ are parameters dealing with lens radial distortion and need to be recovered from camera calibration. In most cases, the dominant non-linear distortion effect is second order radial distortion. So in practice, the above expression is truncated to second order and only k_1 need to be calibrated.

To recover the radial distortion parameters of a camera, a combination of linear and non-linear techniques are usually employed in the camera calibration procedure. For example, in [78], an algorithm of direct linear transformation (DLT) is employed to extract the initial estimates of the camera parameters and then a non-linear least-squares estimation employing the Levenburg-Marquardt algorithm is applied to refine the intrinsic and extrinsic parameters and compute the distortion parameters. In [173], a two-stage technique for camera calibration

is proposed. In the first stage a linear equation system is built based on the *radial alignment constraint* for computing some intrinsic and extrinsic parameters. The rest of the intrinsic and extrinsic parameters are first estimated by ignoring the lens distortion in the second stage and then the refinement of these parameters and the computation of distortion parameter are completed by using a standard optimization scheme such as steepest descent.

3.7 Conclusion

In this chapter, we have a brief review on the basic materials of image formation, which include the light trajectory in the image formation process, the pinhole and finite aperture camera models, the projection matrix of a camera and the camera calibration. All these materials are basic in the field of computer vision but they are very essential to our work. In solving multiview stereo with a small baseline, we build our work on an off-axis aperture camera. For this camera, we introduce an image formation model which combines the stereo and defocus cues in one imaging device. The image formation model and the closed form computation of it is one of the major subcontributions of this system. In solving multiview stereo with a large baseline, to test and evaluate the proposed methods, we generate a few groups of synthetic data. No matter the introduce of the image formation model for the off-axis aperture camera or the generation of the synthetic data, they are all based on a good understanding on the image formation procedure. Although the camera calibration is not a major task in this work, the real data involved in the experiments in our work are all calibrated. So we also include a brief description on camera calibration in this chapter.

Understanding image formation procedure helps us to achieve the task of 3D shape reconstruction. Another issue we need to deal with before we set out the work of reconstruction is choosing a proper way to represent the reconstructed surface. In our work, we use level sets to represent the surfaces and in one of our

methods the algorithm is formulated in a level set framework. The relevant materials of level set methods including its advantages for 3D shape reconstruction are given in the next chapter.

Chapter 4

The Level Set Methods and a Primer on Surface Evolution Methods

4.1 Introduction

As discussed in Section 2.3.1, surfaces can be represented explicitly or implicitly. The level set methods play a very important role in surface evolution where the surfaces are represented implicitly. Some advantages of the level set methods are also described in Section 2.3 in the comparison between the Lagrangian and Eulerian surface deformation methods and between the volume tracking methods and the level set methods.

The level set methods are a group of formulations for surface evolution problems based on a particular class of partial differential equations and the corresponding numerical algorithms for solving the involved partial differential equations. They have been used for tracking the motion of dynamical surfaces in fields including graphics, image processing, tracking, computational dynamics, material science and many others. Rather than an explicit representation in terms of

curves or surfaces as in Lagrangian approaches, in level set methods an interface is represented implicitly through a level set function $\phi(\mathbf{x})$. The interface itself is usually the zero isosurface or zero level set $\phi(\mathbf{x}) = 0$. Various types of surface motion can be described through the evolution of $\phi(\mathbf{x})$ by partial differential equations.

The Chan-Vese algorithm [31] is an example of application of the level set methods to segment objects with smooth boundaries in a given image. It is closely related to the classical Mumford-Shah algorithm [124] but uses a simple level set framework for its implementation. The original formulation of the Chan-Vese algorithm focuses on bi-modal images, which consist of two regions, Ω_{c_i} and Ω_{c_o}, of approximately piecewise constant distinct intensity values c_i and c_o [31]. The bi-modal model can be extended to multimodal images [32].

This chapter is organized as follows: Section 4.2 presents a brief review of some fundaments of level set methods. Numerical methods employed in level set methods are described in Section 4.3. A brief explanation of the marching cubes algorithm is given in Section 4.4, which is followed by the Chan-Vese algorithm in Section 4.5. We tested the effects of some parameters in he Chan-Vese algorithm on behavior of the interface and the convergence to the true segmentation and compared the accuracy of different numerical schemes. The corresponding experimental results are shown with 2D and 3D segmentations in Section 4.6.

4.2 Level Set Methods

The level set method devised by Osher and Sethian [132] provides a way for computing and analyzing the motion of an interface under a velocity field which depends on the position, time, geometry of the interface and external constraints. One of the advantages of the level set method is that, by embedding the interface as the zero level set in a higher-dimensional volume, the level set method can perform numerical computations involving curves and surfaces on a fixed Cartesian

grid without having to parameterize these objects (this is the so-called Euclidean approach). Also, it is easy for the level set method to deal with topological changes such as splitting a shape into two, developing holes, or the reverse of these operations. All these make the level set method a popular tool in computer vision as well as in modeling time-varying objects, like inflation of an airbag, or a drop of ink diffusing in water.

4.2.1 Level Sets

To get a good understanding of level set methods, it is necessary to have a good understanding of what level sets are. Let us consider the example function $z = f(x, y) = (x^2 + y^2) - 4$, where x and y are real numbers. The level sets of $f(x, y)$ are the sets on which the function is constant. Geometrically, a level plane $z =$ constant will cut through the surface $z = f(x, y)$ on a level set. For example, if we let plane $z = 0$ cut through the surface $z = f(x, y)$, the intersection curve is $x^2 + y^2 = 4$. This is the zero level set of the function $z = f(x, y)$. If we let plane $z = 4$ cut through the surface $z = f(x, y)$, the projection of the intersection curve on the 'xy' plane is $x^2 + y^2 = 8$. This is the level set with value 4 of the function $z = f(x, y)$ (Figure 4.1).

In level set methods, the zero level set is usually used to represent the interface, but there is nothing special about the zero level set. Indeed any level set can be translated to the zero level set easily. For example, if we let $z = 1$ cut through the surface $z = f(x, y)$, the level set obtained is $x^2 + y^2 = 5$, which is exactly the zero level set of function $w = g(x, y) = (x^2 + y^2) - 5$. In general, for any function $\hat{\phi}(\mathbf{x})$ and arbitrary level set $\hat{\phi}(\mathbf{x}) = a$ for some scalar $a \in \mathbb{R}$, it is possible to define $\phi(\mathbf{x}) = \hat{\phi}(\mathbf{x}) - a$ so that the zero level set of $\phi(\mathbf{x})$ is identical to the $\hat{\phi}(\mathbf{x}) = a$ level set of $\hat{\phi}(\mathbf{x})$. It can be seen that the only difference between $\hat{\phi}(\mathbf{x})$ and $\phi(\mathbf{x})$ is a scalar translation a, so the function ϕ and $\hat{\phi}$ have the identical partial derivatives and many other similar properties.

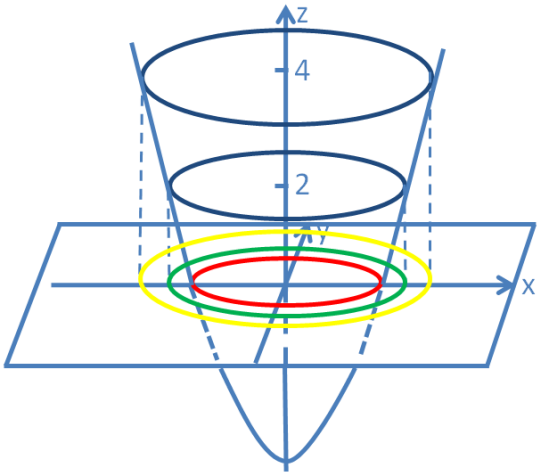


Figure 4.1: An example of 2D level sets in a 3D space. The red circle is the zero level set, the green one is the level set with value 2 and the yellow one is the level set with value 4. See details of the level set function in text.

4.2.2 The Level Set Equation

In level set methods, the surface of interest is embedded implicitly as the zero level set in a higher-dimensional function $\phi(\mathbf{x})$. The problem arising is how to guarantee that the surface is given by the zero level set of the time-dependent level set function $\phi(\mathbf{x})$ at any time. The answer can be obtained through the derivation of the level set equation given by Osher and Sethian [132].

In order to derive an equation of the motion of the level set function $\phi(\mathbf{x})$ and match its zero level set with the evolving surface, it is first required that the level set value of a particle on the surface with path $\mathbf{x}(t)$ be zero (t is the time variable). And hence

$$\phi(\mathbf{x}(t), t) = 0, \forall t \in [0, \infty). \tag{4.1}$$

To find the derivatives with respect to t by the chain rule, we have

$$\phi_t + \nabla\phi(\mathbf{x}(t), t) \cdot \frac{\partial \mathbf{x}}{\partial t} = 0. \tag{4.2}$$

Here, $\nabla\phi$ is the gradient of ϕ, $\frac{\partial \mathbf{x}}{\partial t} = \left[\frac{\partial x_1}{\partial t} \ \frac{\partial x_2}{\partial t} \ \frac{\partial x_3}{\partial t}\right]^T$ is the velocity of point $\mathbf{x}$.

Suppose F supplies the speed in the outward normal direction, then $\frac{\partial \mathbf{x}}{\partial t} \cdot \mathbf{N} = F$, where $\mathbf{N} = \nabla\phi(\mathbf{x})/|\nabla\phi(\mathbf{x})|$ is the normalized normal vector at point $\mathbf{x}$. This yields an evolution equation for ϕ, namely

$$\phi_t + F|\nabla\phi| = 0 \tag{4.3}$$

Here we suppose the initialization of the level set function $\phi(\mathbf{x}, t = 0)$ is given. A way to initialize the level set equation is given in the next section.

This is the level set equation given by Osher and Sethian [132], which describes the time evolution of the level set function in such a way that the zero level set of this evolving function is always identified with the propagating interface.

4.2.3 Signed Distance Function

In multiview stereo, the first step in applying the level set method is to discretize the volume which contains the objects of interest into voxels and initialize the level set function $\phi(\mathbf{x})$ to guarantee that the zero level set is the current surface. A simple way is to set positive values to the voxels outside of the surface, negative values to the voxels inside of the surface and zeros to the voxels on the surface. Also smoothness is a desirable property of the level set function especially when using numerical approximations. A function that satisfies all these properties is the so-called signed distance function. It satisfies $|\nabla\phi(\mathbf{x})| = 1$ and is used to initialize and update the level set function during the evolution of the surface.

Let us define the *distance function* $d(\mathbf{x})$ with respect to a surface $\partial\Omega$ as

$$d(\mathbf{x}) = min(|\mathbf{x} - \mathbf{x_I}|) \quad for \quad all \quad \mathbf{x_I} \in \partial\Omega. \tag{4.4}$$

Notice that $d(\mathbf{x}) = 0$ on the surface where $\mathbf{x} \in \partial\Omega$.

A *signed distance function* is an implicit function ϕ with $|\phi(\mathbf{x})| = d(\mathbf{x})$ for all $\mathbf{x}$ such that $\phi(\mathbf{x}) = d(\mathbf{x}) = 0$ for all $\mathbf{x} \in \partial\Omega$, $\phi(\mathbf{x}) = -d(\mathbf{x})$ for all $\mathbf{x} \in \Omega^-$, and

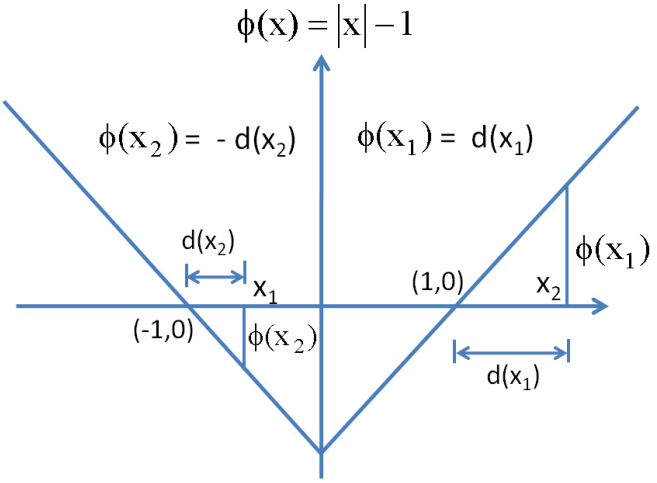


Figure 4.2: An example of signed distance function in 1D case. See details in the text.

$\phi(\mathbf{x}) = d(\mathbf{x})$ for all $\mathbf{x} \in \Omega^+$. Where Ω^- is the volume inside of $\partial\Omega$ and Ω^+ is the volume outside of $\partial\Omega$. In the following sections of this chapter, Ω^- and Ω^+ will be used in the same way.

For example, a signed distance function in the 1D case is $\phi(x) = |x| - 1$ (Figure 4.2). In this example, the surface consists of only two points: $(1, 0)$ and $(-1, 0)$. Points between these two locations are inside of the surface and every other point is outside of the surface. For a point x_1 outside of the surface $\phi(x_1) = d(x_1)$ and for a point x_2 inside of the surface $\phi(x_2) = -d(x_2)$.

Even if the level set function ϕ is initialized as a signed distance function, the evolution of the surface will generally make ϕ drift away from a signed distance. Thus it is always advisable to reinitialize the level set function as an approximate signed distance function occasionally. A signed distance function benefits from analytical simplifications and does not have steep gradients, which is a problem with finite-difference approximations. A reinitialization equation was proposed in [166] in the following form

$$\phi_t + S(\phi_0)(|\nabla\phi| - 1) = 0 \tag{4.5}$$

where ϕ_0 is the initial value of ϕ and $S(\phi_0)$ is the sign function given as the following expression:

$$S(\phi_0(\mathbf{x})) = \begin{cases} 1, & \mathbf{x} \in \Omega^+ \\ 0, & \mathbf{x} \in \partial\Omega \\ -1, & \mathbf{x} \in \Omega^- \end{cases} \tag{4.6}$$

It has been pointed out that good results are obtained when $S(\phi_0)$ is numerically smeared out [166]. So the authors suggest that

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + (\Delta x)^2}} \tag{4.7}$$

where Δx is the size of the Cartesian mesh cell in the corresponding numerical scheme, acting as a smoothing parameter. In fact Δx can be replaced by any small constant value.

The above approach works well when ϕ_0 is neither too flat or too steep near the interface (the zero level set). Otherwise there would be some problems. First, when ϕ_0 is too flat, the quantity of the right hand side of Equation (4.7) is small and the propagating speed in the reinitialization would be slow too. More steps will be needed to reset ϕ_0 to a signed distance function. Second, when ϕ_0 is very steep near the interface, this approach might change the sign of ϕ_0, thus moving the interface across grid points. More detailed analysis can be found in [134]. So the authors of [134] suggest the following scheme which can solve the above mentioned problems:

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + |\nabla\phi_0|^2(\Delta x)^2}} \tag{4.8}$$

In our experiments, we use the last scheme to reinitialize the level set functions when it is necessary.

4.3 Numerical Schemes

In the level set methods, as the surface evolves in time, the values of the level set function at each point evolves too. The level set methods require choosing a good numerical scheme to update the values of the function at each point over small increments of time.

As pointed out in [147], the level set Equation (4.3) can be accurately approximated by computational schemes which exploit techniques borrowed from the numerical solutions of hyperbolic conservation laws. For example, 2D schemes may be developed by using a discrete grid in the $x-y$ domain and substituting finite difference approximations for spatial and temporal derivatives in the following form

$$\frac{\phi_{ij}^{n+1}-\phi_{ij}^{n}}{\Delta t}+F|\nabla\phi_{ij}^{n}|=0 \tag{4.9}$$

where ϕ_{ij}^{n} is the approximation to the solution $\phi(i\Delta x, j\Delta y, n\Delta t)$ with Δx and Δy as the spatial steps and Δt as the time step.

In the above equation, a forward difference scheme in time has been used, and $|\nabla\phi_{ij}^{n}|$ represents some appropriate finite difference operator for the spatial derivative such as the upwind scheme, which has the advantage of preserving cusps and corners. For a higher degree of accuracy, the *essentially nonoscillatory* (ENO) and *weighted essentially nonoscillatory* (WENO) schemes [131] could be chosen. The upwind scheme and ENO and WENO schemes will be discussed in the following sections.

4.3.1 Upwind Scheme

Hyperbolic partial differential equations describe wave propagation problems, for example waves in water, gas, plasmas, traffic flow, etc. In a number of simple cases the wave moves along with unchanged form with a certain speed. The simplest hyperbolic equation exhibiting solutions of this sort is the first order

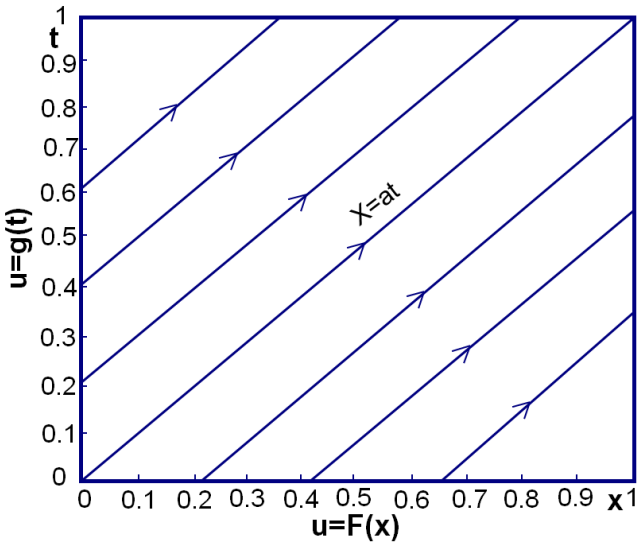


Figure 4.3: The solution of the constant coefficient advection equation is constant along each characteristic line.

advection equation for $u(x,t)$

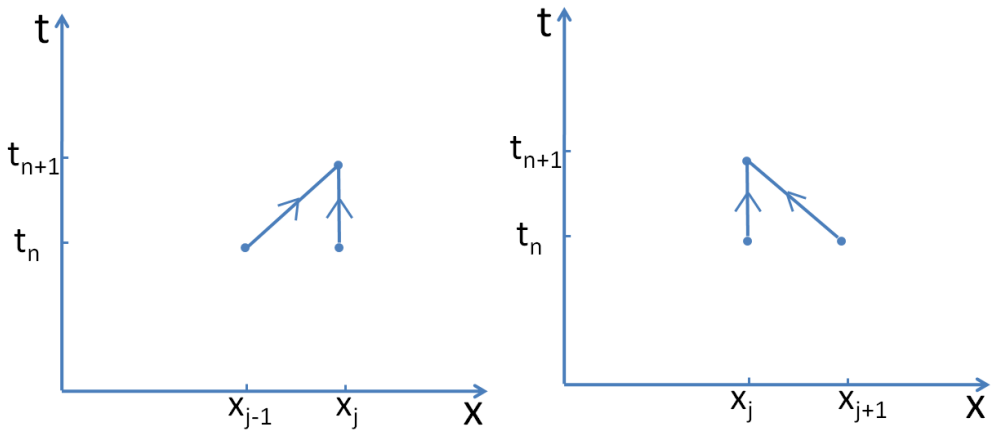
$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \quad with \quad u(x,0) = F(x) \tag{4.10}$$

where a is a constant indicating the wave speed or the velocity of propagation and F is some initial boundary condition. If $a > 0$, the wave moves to the right and if $a < 0$ it moves to the left. If we specify $u(0,t) = g(t)$ when $a > 0$, the exact solution for Equation (4.10) is

$$u(x,t) = \begin{cases} g(t - x/a), & x < at \\ F(x - at), & x > at \end{cases} \tag{4.11}$$

The solution is constant along each characteristic line with slope $dx/dt = a$ as illustrated in Figure 4.3.

It can be seen from the method of characteristics [74] that if the velocity of propagation a satisfies $a > 0$, we should look to the left of a point x_i to determine what value of u will land on the point x_i at the end of a time step. Similarly,



(a) FTBS scheme (b) FTFS scheme

Figure 4.4: In the FTBS scheme, the information travels to the right like in Equation 4.10 with wave speed $a > 0$. In the FTFS scheme, the information travels to the left like in Equation 4.10 with wave speed $a < 0$.

if $a < 0$, the values of u are moving from right to left and we should look to the right to determine an appropriate value of u at point x_i at the end of a time step. The wave direction (i.e., sign(a)) is important for the analytic solution of the advection equation. It is necessary to consider how this "feeds" into the numerical finite difference schemes.

In Figure 4.4, the forward time and backward space (FTBS) and the forward time and forward space (FTFS) schemes are shown. FTBS means that the temporal derivatives are approximated by forward finite differences and the spatial derivatives are approximated by backward finite differences. Similarly FTFS means that the temporal derivatives and the spatial derivatives are all approximated by forward finite differences. Figure 4.4 suggests that if the velocity of propagation $a > 0$, then the spatial derivatives should be approximated by backward finite differences and if $a < 0$, the spatial derivatives should be approximated by forward finite differences. This method of choosing an approximation to the spatial derivatives by biasing the finite difference stencil in the direction where the characteristic information is coming from is known as the upwind scheme.

The combination of the forward temporal finite difference with the upwind difference scheme is a consistent finite difference approximation to the partial differential Equation (4.10), since the approximation error can converge to zero as $\Delta t \rightarrow 0$ and $\Delta x \rightarrow 0$. Here, Δx, Δt are the spatial and temporal steps respectively.

According to the Lax-Richtmyer equivalence theorem [162] a finite difference approximation to a linear partial differential equation is convergent if and only if it is both consistent and stable. The stability of the above scheme can be enforced using the Courant-Friedreichs-Lewy condition (CFL condition), which asserts that the numerical waves should propagate at least as fast as the physical waves, i.e., $\Delta x/\Delta t > |a|$. If the velocity of propagation a varies from node point to node point, the CFL temporal step restriction is

$$\Delta t < \frac{\Delta x}{max|a|} \tag{4.12}$$

where $max|a|$ is chosen to be the largest value of $|a|$ over the entire Cartesian grid.

4.3.2 ENO and WENO Schemes

Upwind scheme has first-order degree accuracy. If a higher order degree accuracy is expected, then ENO (essentially nonoscillatory) schemes and WENO (weighted essentially nonoscillatory) scheme are used in the numerical approximation.

How do ENO and WENO schemes work? Suppose we want to solve the following partial differential equation:

$$\phi_t + \mathbf{a} \cdot \nabla \phi = 0 \tag{4.13}$$

where the t subscript denotes a temporal partial derivative in the time variable t. ∇ is the gradient operator and $\mathbf{a}$ is the given external velocity field.

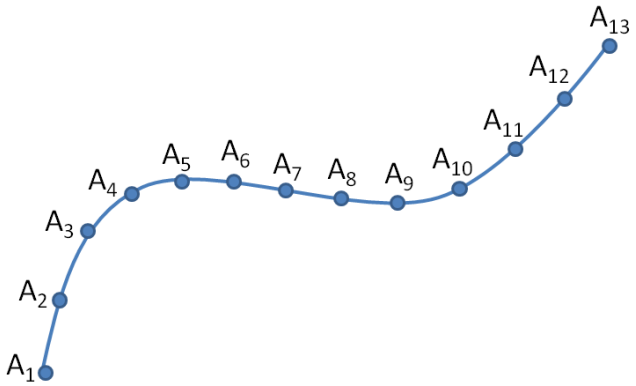


Figure 4.5: A solution discretized by node points.

For simplicity and without losing the generality, the one-dimensional version is considered in the following part of this section, and then $\mathbf{a} \cdot \nabla\phi = a\phi_x$. If ϕ is evolved in time, a rather simple first-order accurate method is the forward Euler method given by

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + a^n \phi_x^n = 0 \tag{4.14}$$

where Δt is the time increment. The superscripts n and $n+1$ denote the time steps. Given the solution of ϕ at time step n, to obtain the solution at time step $n+1$, the ϕ_x need to be approximated at each node point. The more accurate is ϕ_x, the more accurate is the solution. ENO and WENO are the numerical schemes used to approximate ϕ_x.

Suppose a solution of ϕ at time step n is discretized and the nodes $\mathbf{A}_i$ with coordinates $(x_i, \phi(x_i))$ are obtained, where i is an integer number from 1 to the total number of nodes (Figure 4.5). Let us approximate ϕ_x at the node point $\mathbf{A}_5$ as an example. If the ENO1 scheme is used to find ϕ_x at node point $\mathbf{A}_5$, it is possible to choose the combination of $\mathbf{A}_5$ and $\mathbf{A}_4$ or $\mathbf{A}_5$ and $\mathbf{A}_6$ to do the Newton polynomial interpolation and differentiate the polynomial to get ϕ_x. Because only two points are involved in the interpolation, the result is actually a straight line passing through $\mathbf{A}_5$ and $\mathbf{A}_4$ or $\mathbf{A}_5$ and $\mathbf{A}_6$. If the line passing through $\mathbf{A}_5$ and

$\mathbf{A}_4$ is used, it is the same as using the forward time and backward space (FTBS) scheme in upwind scheme. If the line passing through $\mathbf{A}_5$ and $\mathbf{A}_6$ is used, it is the same as using the forward time and forward space (FTFS) scheme in upwind scheme. Recall that the upwind scheme determines which combination of the node points should be used. In fact, ENO1 is exactly the upwind scheme.

When the ENO2 scheme is used to find ϕ_x at node point $\mathbf{A}_5$. If the combination of $\mathbf{A}_4$ and $\mathbf{A}_5$ is chosen to approximate ϕ_x at $\mathbf{A}_5$ in the ENO1 scheme, and then there are two sets of points that can be chosen in the ENO2 scheme, i.e., the combination of $\mathbf{A}_3$, $\mathbf{A}_4$ and $\mathbf{A}_5$ or the combination of $\mathbf{A}_4$, $\mathbf{A}_5$ and $\mathbf{A}_6$. The nonoscillatory property is the criterion used to decide which node combinations to choose. The basic idea of the nonoscillatory property is to use the smoothest possible polynomial interpolation to find ϕ and then differentiate to get ϕ_x. To reconstruct a polynomial based on some node points, it is necessary to have a review of the concept of the divided differences. If the mesh spacing is assumed uniform with step Δx, the zeroth divided differences of ϕ are defined at each grid node i (located at x_i) as:

$$D_i^0\phi = \phi_i \tag{4.15}$$

The first divided differences of ϕ are defined midway between grid nodes as:

$$D_{i+1/2}^1\phi = \frac{D_{i+1}^0\phi - D_i^0\phi}{\Delta x} \tag{4.16}$$

The second divided differences are defined at the grid nodes as:

$$D_i^2 = \frac{D_{i+1/2}^1\phi - D_{i-1/2}^1\phi}{2\Delta x} \tag{4.17}$$

The third divided differences are defined midway between the grid nodes as:

$$D_{i+1/2}^3\phi = \frac{D_{i+1}^2\phi - D_i^2\phi}{3\Delta x}. \tag{4.18}$$

To find ϕ_x at node point $\mathbf{A}_5$ with ENO2 scheme, if the combination of the node points $\mathbf{A}_3$, $\mathbf{A}_4$ and $\mathbf{A}_5$ is used to reconstruct the polynomial, then the polynomial has the following form:

$$\phi(x) = D_5^0\phi + D_{4+1/2}^1\phi(x - x_5) + D_4^2\phi(x - x_4)(x - x_5). \tag{4.19}$$

If the combination of the node points $\mathbf{A}_4$, $\mathbf{A}_5$ and $\mathbf{A}_6$ is used to reconstruct the polynomial and then the polynomial will be changed into the following form:

$$\phi(x) = D_5^0\phi + D_{5+1/2}^1\phi(x - x_5) + D_5^2\phi(x - x_5)(x - x_6) \tag{4.20}$$

The values of $|D_4^2\phi|$ and $|D_5^2\phi|$ indicate which of the polynomial interpolants has more variation. The aim is to avoid interpolating near large variations such as discontinuities or steep gradients. Thus, if $|D_4^2\phi| < |D_5^2\phi|$, the right side of Equation (4.19) will be differentiated to find ϕ_x at node point $\mathbf{A}_5$ otherwise the Equation (4.20) will be used.

If the combination of $\mathbf{A}_5$ and $\mathbf{A}_6$ is chosen to approximate ϕ_x at $\mathbf{A}_5$ in the ENO1 scheme, then the same procedure will be conducted out on the combination of $\mathbf{A}_4$, $\mathbf{A}_5$ and $\mathbf{A}_6$ and the combination of $\mathbf{A}_5$, $\mathbf{A}_6$ and $\mathbf{A}_7$ to find ϕ_x at node point $\mathbf{A}_5$ with ENO2 scheme.

Based on the node points used in the ENO2 schemes, the node points used in the ENO3 schemes are the point sets obtained by adding the next point to the left or the next point to the right. The polynomials are built in the same way as above. For example, if we want to approximate ϕ_x at node point $\mathbf{A}_5$ with the ENO3 scheme, suppose node points $\mathbf{A}_4$, $\mathbf{A}_5$ and $\mathbf{A}_6$ are used in the ENO2 scheme, then the node points used in ENO3 schemes are $\mathbf{A}_3$, $\mathbf{A}_4$, $\mathbf{A}_5$ and $\mathbf{A}_6$ or $\mathbf{A}_4$, $\mathbf{A}_5$, $\mathbf{A}_6$ and $\mathbf{A}_7$. In this case, the values of $|D_{5-1/2}^3\phi|$ and $|D_{5+1/2}^3\phi|$ are compared, and the polynomial corresponding to the smaller one will be used to approximate ϕ_x.

For a certain node combination in ENO1 scheme, two different node combinations can be used in the corresponding ENO2 scheme and three different node combinations can be used in the corresponding ENO3 schemes. For example, if we use $\{\mathbf{A}_4, \mathbf{A}_5\}$ to approximate the value of ϕ_x at node point $\mathbf{A}_5$ with the ENO1 scheme, the two node combinations for ENO2 scheme are $\{\mathbf{A}_3, \mathbf{A}_4, \mathbf{A}_5\}$ and $\{\mathbf{A}_4, \mathbf{A}_5, \mathbf{A}_6\}$ and the three node combinations for ENO3 scheme are: $\{\mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4, \mathbf{A}_5\}$, $\{\mathbf{A}_3, \mathbf{A}_4, \mathbf{A}_5, \mathbf{A}_6\}$ and $\{\mathbf{A}_4, \mathbf{A}_5, \mathbf{A}_6, \mathbf{A}_7\}$. By differentiating the polynomials built upon each node combination in the same as above, three estimates of the value of ϕ_x are obtained. They are denoted as ϕ_x^1, ϕ_x^2 and ϕ_x^3 respectively. In the ENO3 scheme, exactly one of the three is picked. The drawback to do so is that it is overkill in smooth regions where the data are well behaved. To overcome this, a weighted ENO (WENO) is proposed, which is a convex combination of ϕ_x^1, ϕ_x^2 and ϕ_x^3 given by

$$\phi_x = \omega_1\phi_x^1 + \omega_2\phi_x^2 + \omega_3\phi_x^3 \tag{4.21}$$

where the $0 \leq \omega_k \leq 1$ are the weights with $\omega_1 + \omega_2 + \omega_3 = 1$. The key observation of obtaining high-order accuracy in smooth regions is that weights of $\omega_1 = 0.1$, $\omega_2 = 0.6$ and $\omega_3 = 0.3$ give the optimal fifth-order accurate approximation to ϕ_x. The details of the proof can be seen in [85].

A comparison between the accuracy of ENO2 and WENO schemes is given in Section 4.6 with the `Sawteeth` data set.

4.4 Visualizing Isosurfaces: Marching Cubes

Marching cubes is one of the latest algorithms for rendering isosurfaces from 3D data, originally proposed by Lorensen and Cline [111]. The given 3D data are usually a volume discretized in voxels with a scalar value attached to each of them. Each voxel identifies a cube. A cube can be built by connecting the centers

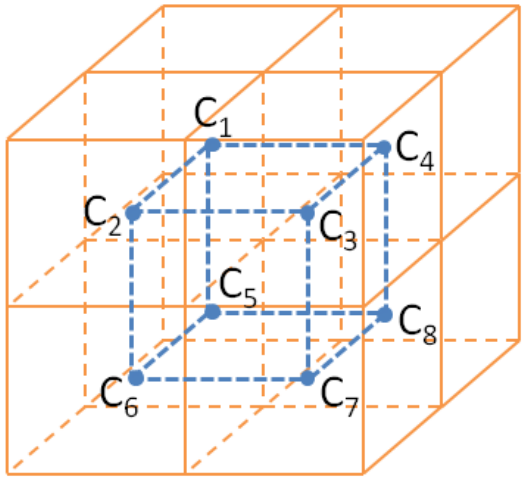


Figure 4.6: The cube built by connecting the centers of eight neighboring voxels.

of eight neighboring voxels as depicted in Figure 4.6. In this figure, the vertices $\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_8$ of the cube are the centers of eight neighboring voxels. The value at a vertex of a cube is the same as the value of the corresponding voxel.

For a user-specified isovalue and a given cube, if the isovalue is less or greater than the values at all the vertices, then the isosurface with respect to the given isovalue does not cut through this cube. On the other hand, if the isovalue is less than the values at some vertices and at the same time greater than those of others then the corresponding isosurface will go through this cube. An example is depicted in Figure 4.7. In this figure, the values in the brackets behind each vertex letter are the values at the corresponding vertices and the user-specified isovalue is supposed to be 0.5. Because there are some vertices whose values are less than the given isovalue such as $\mathbf{C}_6$ and $\mathbf{C}_7$ and there are some vertices whose values are greater than the given isovalue, so the the isosurface will intersect with this cube. To find the intersection of the isosurface with this cube, we need to find the intersections of the isosurface with some edges of this cube first. If an edge with the value at one end point is less than the isovalue and the value at the other end point is greater than the isovalue, then it will intersect with the isosurface.

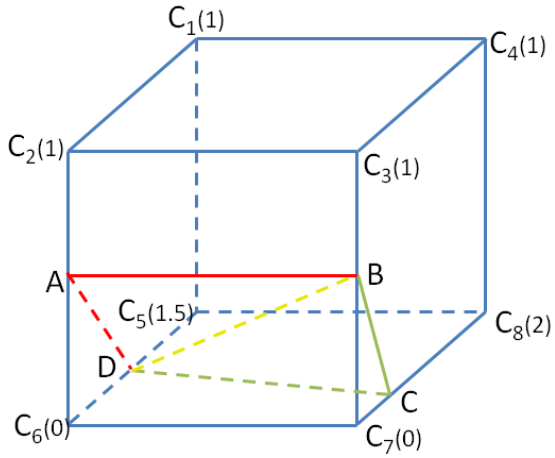


Figure 4.7: One configuration of the intersection of the isosurface with a cube.

Under the given conditions, it can be seen that the isosurface intersects with the edges $\mathbf{C_2C_6}$, $\mathbf{C_3C_7}$, $\mathbf{C_7C_8}$ and $\mathbf{C_5C_6}$. Suppose the isosurface intersects edge $\mathbf{C_2C_6}$ at point $\mathbf{A}$, then the location of point $\mathbf{A}$ on edge $\mathbf{C_2C_6}$ can be decided by a linear interpolation. That is the location of point $\mathbf{A}$ should satisfy the following expression:

$$\frac{|C_2A|}{|AC_6|} = \frac{v_2 - v_{iso}}{v_{iso} - v_6} \tag{4.22}$$

where $|\mathbf{C_2A}|$ and $|\mathbf{AC_6}|$ are the lengths of the corresponding line segments, v_2 and v_6 are the values at $\mathbf{C_2}$ and $\mathbf{C_6}$ respectively and v_{iso} is the given isovalue. In this example, because $v_2 = 1$, $v_6 = 0$ and $v_{iso} = 0.5$, it is easy to find that $\mathbf{A}$ is the midpoint of edge $\mathbf{C_2C_6}$ i.e., $|\mathbf{C_2A}| = |\mathbf{AC_6}|$. In the same way the locations of the intersecting points $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ can be found on the corresponding edges. According to the values at the relevant vertices, we have that $|\mathbf{C_3B}| = |\mathbf{BC_7}|$, $|\mathbf{C_7C}| = 1/4 \cdot |\mathbf{CC_8}|$ and $|\mathbf{C_6D}| = 1/3 \cdot |\mathbf{DC_5}|$. Thus two triangles $\triangle\mathbf{ABD}$ and $\triangle\mathbf{BCD}$ are generated for rendering the isosurface.

Due to different combinations of the values at the vertices of a cube, there are 15 different configurations of the intersections of the cubes with the isosurface (Figure 4.8). More details can be found in [111]. After the algorithm determines how the surface intersects a cube, then it will move to the next cube. When all

the intersection triangles have been generated, they will be fused into a surface.

The final step in marching cubes calculates a unit normal for each triangle vertex. The rendering algorithms use these normals to produce Gouraud-shaded images. In Figure 4.9, 8 neighboring cubes and the corresponding coordinate system are displayed. In this figure, $\mathbf{C}_i$ $(0 < i < 12)$ are the vertices of the cubes and the values at the corresponding vertices are denoted with v_i. To estimate the normal vectors for triangle vertices, we can estimate the gradient vectors at the cube vertices first and then using linear interpolation to find the normal vectors for triangle vertices. Next, we use finding the normal vector at vertex $\mathbf{A}$ as an example. To find the normal vector at point $\mathbf{A}$, we need to estimate the gradient vectors at cube vertices $\mathbf{c}_2$ and $\mathbf{c}_6$. The gradient at $\mathbf{C}_2$ can be estimated by central differences along the three coordinate axes by:

$$\begin{aligned} G_{2x} &= \frac{v_9 - v_2}{2\Delta x} \\ G_{2y} &= \frac{v_{10} - v_2}{2\Delta y} \\ G_{2z} &= \frac{v_{11} - v_2}{2\Delta z} \end{aligned} \tag{4.23}$$

where$(G)_2 = [G_{2x}\ G_{2y}\ G_{2z}]^T$ is the gradient vector at cube vertex $\mathbf{C}_2$ and $\Delta x = \Delta y = \Delta z$ are the length of an edge of the cube.

In the same way, the gradient vector $\mathbf{G}_6$ at the cube vertex $\mathbf{C}_6$ can be estimated. And then the normal vector $\mathbf{G}_A$ at the triangle vertex $\mathbf{A}$ can be computed by:

$$\mathbf{G}_A = \frac{|\mathbf{C}_2\mathbf{A}| \cdot \mathbf{G}_6 + |\mathbf{C}_6\mathbf{A}| \cdot \mathbf{G}_2}{\Delta x} \tag{4.24}$$

Throughout this thesis, all the 3D surfaces for displaying experimental results are generated with the marching cubes method. In the next section, we will give an algorithm for 2D and 3D segmentation. This is an concrete example using the marching cubes method to generate the 3D surface.

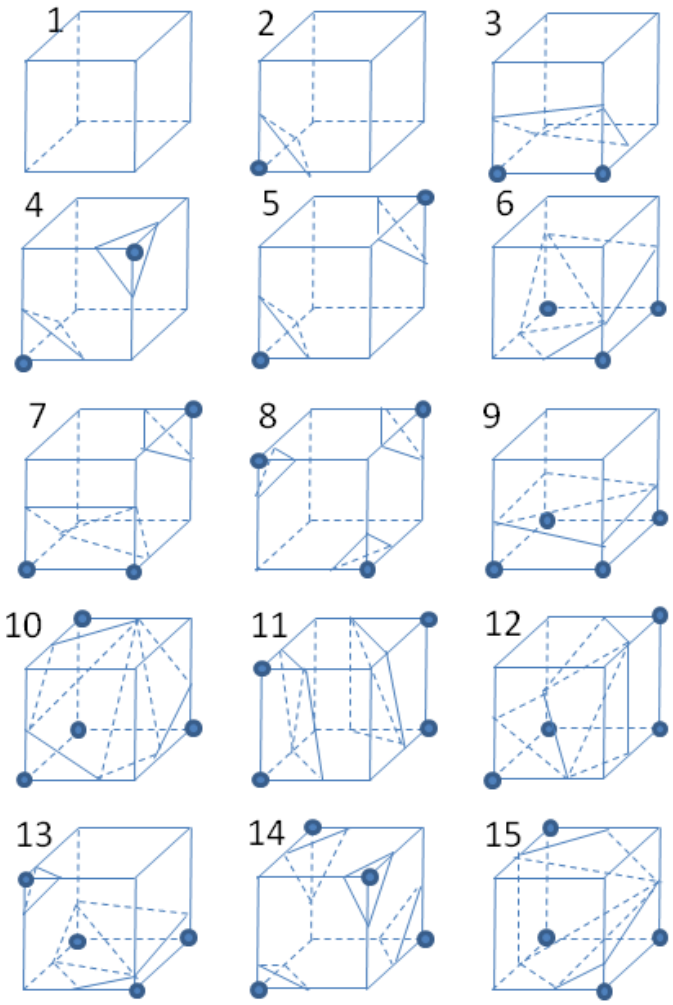


Figure 4.8: The 15 different configurations of a cube with a isosurface. If a vertex is denoted with a black dot, it means that the corresponding value at this vertex is less than the given isovalue. Otherwise, the corresponding value is greater than the given isovalue.

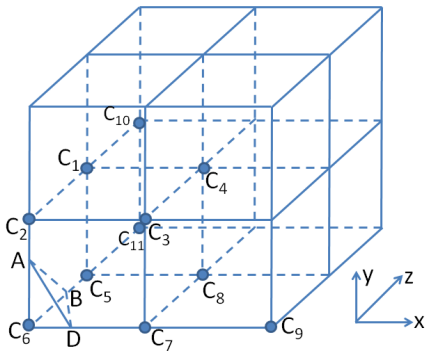


Figure 4.9: Eight neighboring cubes and the corresponding coordinate system. $\triangle$**ABC** is the intersection of the isosurface with the relevant cube.

4.5 The Chan-Vese Algorithm

The Chan-Vese algorithm [31, 32] is an active contour algorithm based on Mumford-shah segmentation techniques and the level set methods. In this model, the evolving curve does not depend on an edge-function to stop on the desired boundary. So it can detect objects whose boundaries are not necessarily defined by gradients, for which the classical active contour models are not applicable. The general form of the Chan-Vese model is

$$\begin{aligned} F(c_1, c_2, C) &= \mu \cdot Length(C) + \nu \cdot Area(inside(C)) \\ &+\lambda_1 \int_{inside(C)} |u_0(x,y) - c_1|^2 dxdy \\ &+\lambda_2 \int_{outside(C)} |u_0(x,y) - c_2|^2 dxdy \end{aligned} \tag{4.25}$$

where u_0 is the image which is formed by two regions of approximatively piecewise-constant intensities, with distinct values u_0^i and u_0^o. C is a closed curve and the constants c_1, c_2 are the averages of u_0 inside C and outside C respectively. $\mu, \nu \geq 0$, $\lambda_1, \lambda_2 > 0$ are the weight parameters.

In Equation (4.25), the first two terms on the right hand side are the regularization terms based on the length of the curve C or the area of the part inside of

curve C. The last two terms on the right hand side are the data fidelity terms. These terms guarantee that only when C is the boundary between the two regions u_0^i and u_0^o Equation (4.25) obtains its minimum value.

The corresponding level set formulation of the Chan-Vese model is

$$\begin{aligned} F(c_1, c_2, \phi) &= \mu \int_\Omega \delta(\phi(x,y))|\nabla\phi(x,y)|dxdy \\ &+\nu \int_\Omega H(\phi(x,y))dxdy \\ &+\lambda_1 \int_\Omega |u_0(x,y) - c_1|^2 H(\phi(x,y))dxdy \\ &+\lambda_2 \int_\Omega |u_0(x,y) - c_2|^2 (1 - H(\phi(x,y)))dxdy \end{aligned} \tag{4.26}$$

where Ω is the image domain. ϕ is an unknown function which defines the evolving curve C as the zero level set of it. H is the Heaviside function and δ is the one-dimensional Dirac delta function defined by

$$H(z) = \begin{cases} 1, & if \quad z \geq 0 \\ 0, & if \quad z < 0, \end{cases} \qquad \delta(z) = \frac{d}{dz}H(z) \tag{4.27}$$

The Euler-Lagrange equation [87] associated to Equation (4.26) for ϕ is

$$\begin{aligned} &\frac{\partial\phi}{\partial t} = \delta_\epsilon(\phi)\left[\mu div\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \nu - \lambda_1(u_0 - c_1)^2 + \lambda_2(u_0 - c_2)^2\right] = 0 \\ &in \quad (0,\infty)\times\Omega, \\ &\phi(\mathbf{0}, x, y) = \phi_0(x,y) \quad in \quad \Omega, \\ &\frac{\delta_\epsilon(\phi)}{|\nabla\phi|}\frac{\partial\phi}{\partial\vec{n}} = 0 \quad on \quad \partial\Omega \end{aligned} \tag{4.28}$$

where $\vec{n}$ denotes the exterior normal to the boundary $\partial\Omega$, and $\partial\phi/\partial\vec{n}$ denotes the normal derivative of ϕ at the boundary. δ_ϵ is the regularized version of the Dirac delta function (see Equation (4.29)).

4.6 Experimental Results of 2D and 3D Segmentation with the Chan-Vese Approach

In this section, some experimental results of 2D and 3D segmentation with the Chan-Vese algorithm are shown. At the same time, the effects of the parameter ϵ on the convergence of the algorithm are studied. A comparison on the accuracy of the ENO2 and WENO schemes is also shown with the `Sawteeth` data set.

In all the experiments, the area term (i.e., the second term on the right hand side of Equation (4.26)) is omitted. There are two reasons to do so: first, using the length of the zero level set curve for the regularization is more powerful than using the area term, because the length term is working on the zero level set directly. Second, keeping the area term leads to more computation which affects the efficiency of the whole algorithm. A regularized version of the Heaviside function has the following form

$$H_\epsilon(z) = \frac{1}{2}\left(1 + \frac{2}{\pi} arctan\left(\frac{z}{\epsilon}\right)\right) \tag{4.29}$$

where ϵ is the parameter controlling the degree of the regularization. As $\epsilon \to 0$, the above function converges to the Heaviside function.

There are two groups of experiments shown in the 2D case. In the first group, the image used for segmentation is called "YinYang-Fish" with size 217×233 pixels (see Figure 4.10).

The parameters in Equation (4.26) are set in the following way: first, because we have decided to omitted the area term, so in all the experiments $\nu = 0$. Second, because we do not want favor any of the two data terms, so λ_1 and λ_2 should be equal to each other. And then the ratio between μ and λ_1 or λ_2 tunes the work of the regularization term against the data terms. In all the experiments, we set $\lambda_1 = \lambda_2 = 1$, and the only parameter need to be tuned is μ. With a larger value setting to μ, the regularization term will do more work. In the experiments of

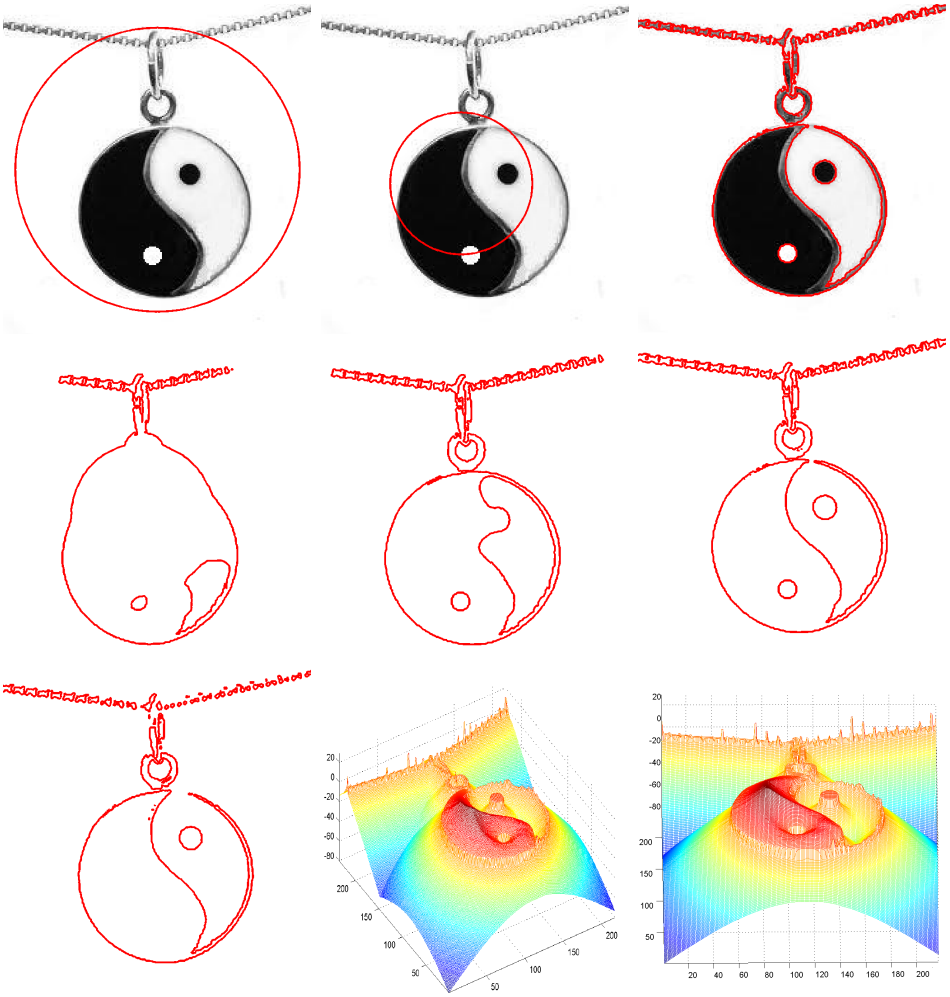


Figure 4.10: The segmentation of `YinYang-Fish`. The first two images in the first row show the original image with two different initial zero level set curves from ϕ_{01} and ϕ_{02}. The third image shows the final segmentation. The images in the second row show the zero contours of the level set function after different iterations. The first image in the last row shows that when the value of ϵ is too small the eye of the black fish can not be detected and the last two images are two views of the final level set function. See details in the text.

"YinYang-Fish" dataset displayed in Figure 4.10, we set $\mu = 20$. The effect on regularization by setting different values to μ is displayed in Figure 4.11 with the `Sawteeth` dataset.

One more parameter need to be tuned in this algorithm is the parameter ϵ in Equation 4.29. The corresponding regularized Dirac delta function is $\delta_\epsilon(z) =$

$\epsilon/[\pi(\epsilon^2 + z^2)]$, which is non-zero everywhere. This property makes the Euler-Lagrange equation of ϕ act on all level curves so that the algorithm tends to compute a global minimizer. Notice that to take advantage of this property the value of ϵ should not be too small.

In Figure 4.10, we show the results from two different initial level set equations. One of the initial level set equations is:

$$\phi_{01}(x, y) = 80 - \sqrt{(x - 109)^2 + (y - 117)^2} \tag{4.30}$$

and the other one is:

$$\phi_{02}(x, y) = 50 - \sqrt{(x - 99)^2 + (y - 127)^2} \tag{4.31}$$

It can be seen from the first two images in Figure 4.10, the initial zero level contour from the level set function ϕ_{01} is much larger than that from the level set function ϕ_{02}. To detect the desired contour, they need to evolve in almost opposite way i.e., one need to shrink and the other one need to expand. In both experiments, if we set $\epsilon > 3$ then the results from both experiments are the same as shown with the last image in the second row in Figure 4.10. But if ϵ is set less than 2, when the level set equation is initialized as ϕ_{01}, the white eye of the black fish in the image can not be detected (see the first image of th third row displayed in Figure 4.10). Based on the results of a series of experiments, we can draw the conclusion that although the Chan-Vese algorithm is not a convex formulation, but by setting ϵ with a relative larger value, say 5, it seems there is no problem to obtain the global minimizer.

The second group of experiments in the 2D case are done with a group images called `Sawteeth` data set, which are images of a polygon with one side shaped like sawteeth (see the first image in Figure 4.11). The image has a noisy background with size 300 $\times$ 360 pixels. With this group of experiments, first, we want to

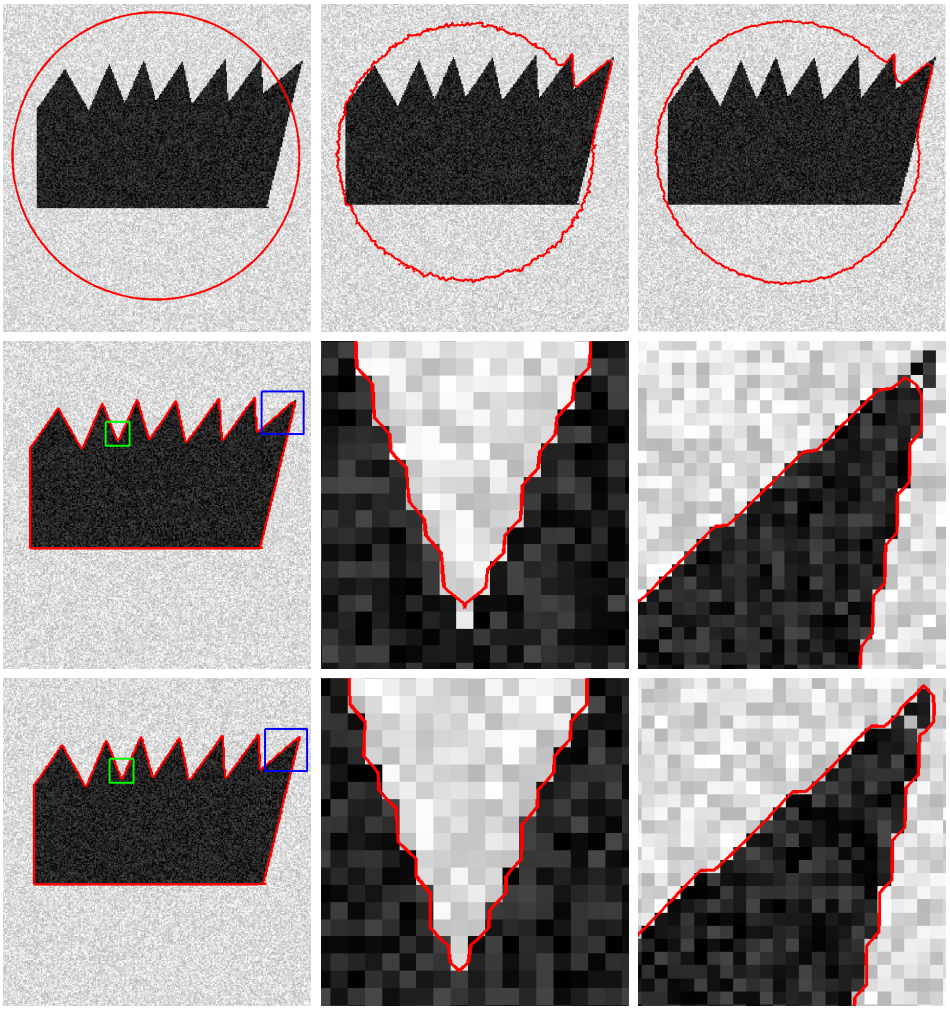


Figure 4.11: The segmentation of `Sawteeth`. The first image in the first row shows the original image with the initial zero level set curve from ϕ_{03}. The last two images in the first row show effect of the strength of the regularization term on the smoothness of the zero level set contours. From the third image we can see: the larger the regularization term, the smoother the zero level set curve. The images in the second row and third row show the segmentation results from ENO2 and WENO schemes. The small windows in the green and blue squares in the first images in each row are enlarged in the second and third images. The WENO scheme yields a more accurate boundary approximation than the ENO2 scheme.

show the different effects of the regularization term by setting different values to μ. Second, we want to show the different accuracies of ENO and WENO schemes.

The initial level set function in this group of experiments is:

$$\phi_{03}(x, y) = 120 - \sqrt{(x - 180)^2 + (y - 150)^2} \tag{4.32}$$

In the experiments shown with the second and third images displayed in the first row in Figure 4.11, all the other parameters are set the same, the only parameter changed is μ, which controls the strength of the regularization term. For the second image, $\mu = 1$ and for the third image $\mu = 20$. Clearly, with more regularization strength in the third image, the zero level set curve is smoother than that in the second image. That is the larger is the value of μ the more strength the regularization term has.

Another purpose of this group of experiments is to show the difference of the accuracy of different numerical schemes. From Section 4.3, it has been known that the ENO1 scheme, i.e., the upwind scheme, has first order accuracy, ENO2 scheme has second order accuracy, ENO3 scheme has third order accuracy and WENO scheme has fifth order accuracy. In Figure 4.11, the segmentation results from ENO2 scheme and WENO scheme are shown in the second row and third row respectively. The first images in each row show the segmentation results. In each image, there are two small windows included in a green and blue squares, which are enlarged in the second and third images in each row respectively. With the enlarged windows in both rows, one can visually appreciate that the WENO scheme is more accurate than the ENO2 scheme.

In the 3D case, we display two groups of experiments. In these experiments, the same regularized vision of the Heaviside function as in the 2D cases is used.

In Figure 4.12, the segmentation of two cubes in a volume is shown. In these experiments, the volume size is $50 \times 50 \times 50$ voxels. The values at voxels within the objects are set to 1 and the values at the voxels outside of the objects are set

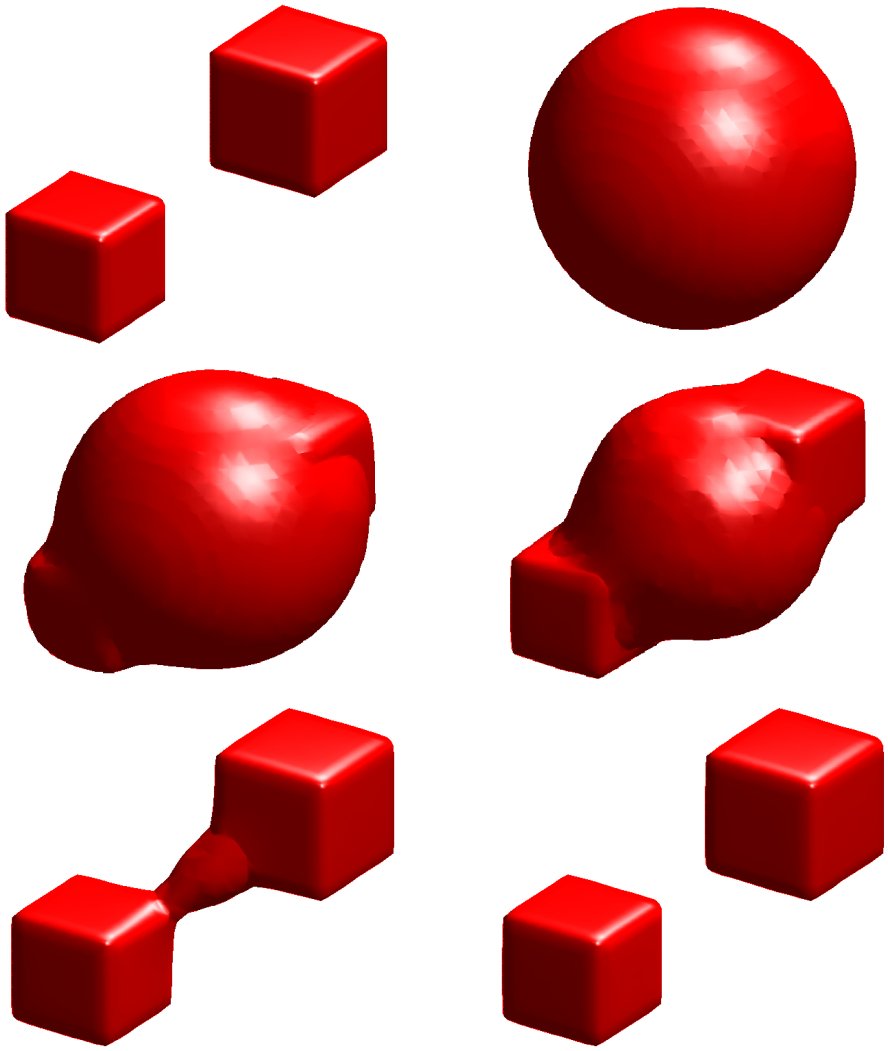


Figure 4.12: The segmentations of `Twocubes` in 3D space. First row: the true objects to be segmented(left) and the initialization of the zero level set(right). Second row: the zero level set after 30 iterations(left) and the zero level set after 90 iterations. Third row: the zero level set after 180 iterations (left) and final segmented objects after 240 iterations.

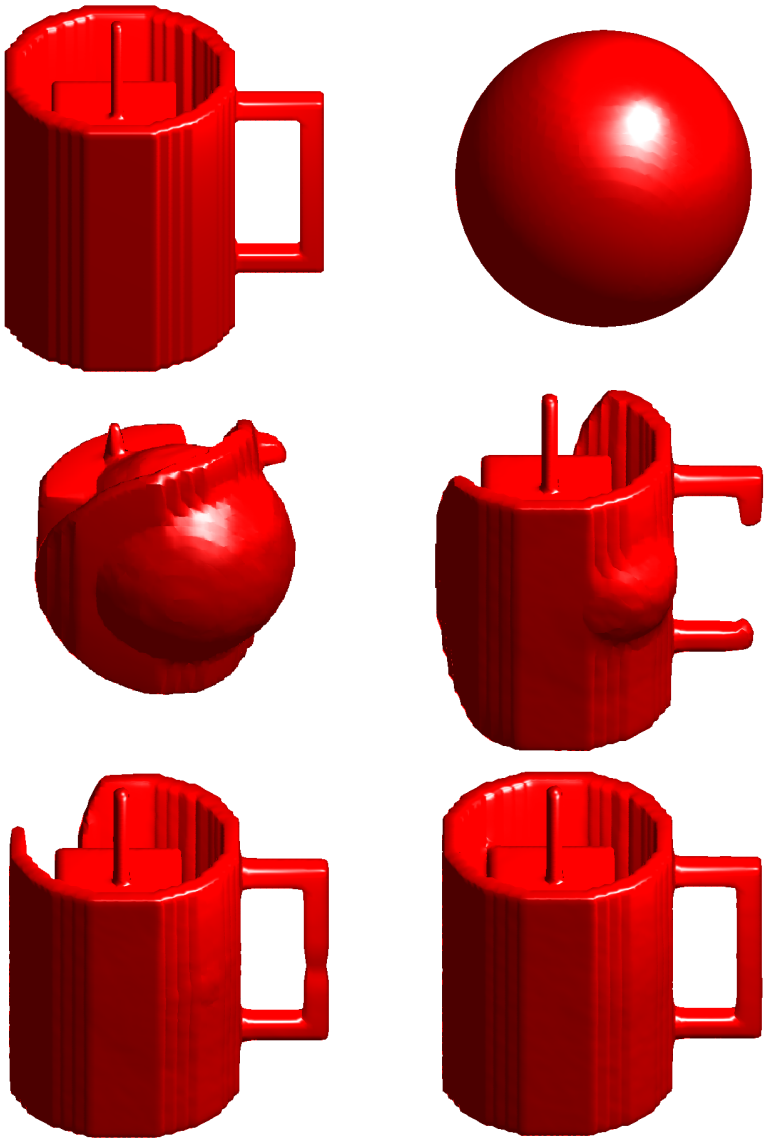


Figure 4.13: The segmentations of `Cupcube` in 3D space. First row: the true objects to be segmented(left) and the initialization of the zero level set(right). Second row: the zero level set after 60 iterations(left) and the zero level set after 180 iterations. Third row: the zero level set after 270 iterations (left) and final segmented objects after 360 iterations.

to -1. 20% zero-mean noise is added. The parameters in Equation 4.26 and 4.29 are set as: $\mu = 5$, $\nu = 0$, $\lambda_1 = \lambda_2 = 1$ and ϵ=5. The ENO2 scheme is used in these experiments. The level set function is initialized as:

$$\phi_{04}(x, y) = 3 \cdot e^{-((x-25)^2+(y-25)^2+(z-25)^2)/200} - 1.2 \tag{4.33}$$

The initial zero level set is a sphere as shown with second image in Figure 4.12. From this group of experiments, it can be seen that the level set methods can deal with topological changes automatically. Compared with the detected result with the ground truth, the two cubes are detected very accurately and only the edges of the detected cubes are slightly smoothed because of the regularization term.

The second group of experiments show the segmentations of a cup containing a cube with a thin bar passing through it (see the first image in Figure 4.13). It is not a easy task to segment an object like this from a volume. First the topology of this object is complicated. Second, there is a thin bar contained in this object. It is always a difficulty to segment a thin object from a volume. But it can be seen from Figure 4.13, we have obtained a very good segmentation result. The only difference we can see between the segmented object and the ground truth is the segmented bar is slightly thicker than the original one. This is because the regularized version of the Heaviside function spreads out the positive values a little. In these experiments, the volume size is $60 \times 60 \times 60$ voxels. The values at the voxels within the objects are set to 1 and the values at the voxels outside of the objects are set to -1. 30% zero-mean noise is added. The parameters in Equation 4.26 and 4.29 are set as: $\mu = 8$, $\nu = 0$, $\lambda_1 = \lambda_2 = 1$ and ϵ=2. The ENO3 scheme is used in these experiments. The level set function is initialized as:

$$\phi_{05}(x, y) = e^{-((x-30)^2+(y-30)^2+(z-30)^2)/200} - 0.2 \tag{4.34}$$

The initial zero level set is also a sphere as shown in the second image in Figure 4.13.

4.7 Conclusion

In this chapter, we first had a review on the basic materials of the level set methods including the concepts of the level set, the signed distance function and the derivation of the level set method proposed by Osher and Sethian, and then the numerical schemes for solving the partial differential equations involved in the level set methods are discussed in more details.

Because the marching cubes method is a popular algorithm for rendering isosurfaces from 3D data and most 3D surfaces displayed in this thesis are generated with the marching cubes method, so how to generate a 3D surface with the marching cubes method is presented in this chapter. In marching cubes methods, two key steps are how to find the intersections of a isosurface with the edges of a small cube and how to compute the normal vectors at each intersection point. They are all discussed in detail.

The Chan-Vese algorithm is a concrete application of the level set methods. It is also the inspiration of the last algorithm proposed in this work for solving multiview stereo with a large baseline. After an explanation of the fundament of the Chan-Vese algorithm, we showed a series of experimental results of 2D and 3D segmentation. With these experiments, we studied the functions of different parameters in the Chan-Vese algorithm, especially, we discussed the influence of the parameter ϵ in Equation 4.29 on the convergence to a global minimum of the solution.

We also compared the accuracy of different numerical schemes, i.e., ENO and WENO schemes. Our conclusion is that to reconstruct a smoothing surface the difference between them are not significant but to reconstruct an object with sharp corners the difference between then are obvious. In the last situation, we

should use the schemes with higher order accuracy.

The materials discussed in this chapter lay a solid foundation for our work presented in Chapter 6. But in next chapter, we will first present our work for solving multiview stereo with a very small baseline based on an off-axis aperture camera.

Chapter 5

Small-Baseline Multiview Stereo: The Off-Axis Aperture Camera

5.1 Introduction

In this chapter we present a novel 3D surface and image reconstruction method based on the off-axis aperture camera. The key idea is to change the size or the 3D location of the aperture of the camera lens so as to extract selected portions of the light field of the scene. This results in an imaging device that blends defocus and stereo information [33, 53, 76, 142]. In particular, this imaging device can be used in a small-scale space where the camera motion is constrained by the surrounding environment, such as in 3D endoscopy and 3D microscopy [108, 163]. The work presented in this chapter is published in our paper [43].

Off-axis apertures have been used in several fields, such as ophthalmology, astronomy and microscopy [125, 185]. In astronomy, for instance, multi-hole off-axis apertures have been widely used in testing optical elements for nearly one hundred years. A mask with two off-axis apertures, also called Hartmann Mask or Scheiner Disk, was invented to correct of quantified defocus by the Jesuit astronomer Christoph Scheiner (1572-1650). This technique is based on the prin-

ciple that, unless the scene is brought into focus, the two apertures will generate an effect similar to double vision.

Zomet and Nayar [196] proposed a camera consisting of an image detector and a special aperture, but no lens. The aperture is a set of parallel light attenuating layers with controllable transmittances in space and time. Compared with the work of [196], we make use of a lens in our system and therefore our image formation model is very different from theirs. The off-axis camera we use can be thought of as a device composed of an image plane, a thin lens and a moving aperture as depicted in Figure 5.1. The main advantage of this camera is that no lens or image plane movement is required, as well as no relative motion between camera and object is needed. We obtain images by changing the position and diameter of the aperture. While this can be achieved by physically moving a mask in front of the lens (as we do in our experiments), one could also place an LCD transparent display and turn its pixels on or off in space and in time so as to form the desired attenuation mask, as it was done in [196].

This chapter is arranged as follows: Section 5.2 gives a description of the off-axis aperture camera and an image formation model that simultaneously captures defocus and stereo information. As the proposed image formation model involves a non-trivial deformation of the scene, we introduce the concept of scene space rectification. In Section 5.3 we show that it helps the reconstruction of the 3D shape. A gradient flow algorithm for 3D reconstruction and image restoration is derived in Section 5.4. This chapter is terminated with experimental results and a brief conclusion.

5.2 Off-Axis Aperture Camera

In this section, we will give a description of the off-axis aperture camera model we proposed in this work. First, we present the geometry of this camera model, and then the image formation model of this camera will be analyzed in detail.

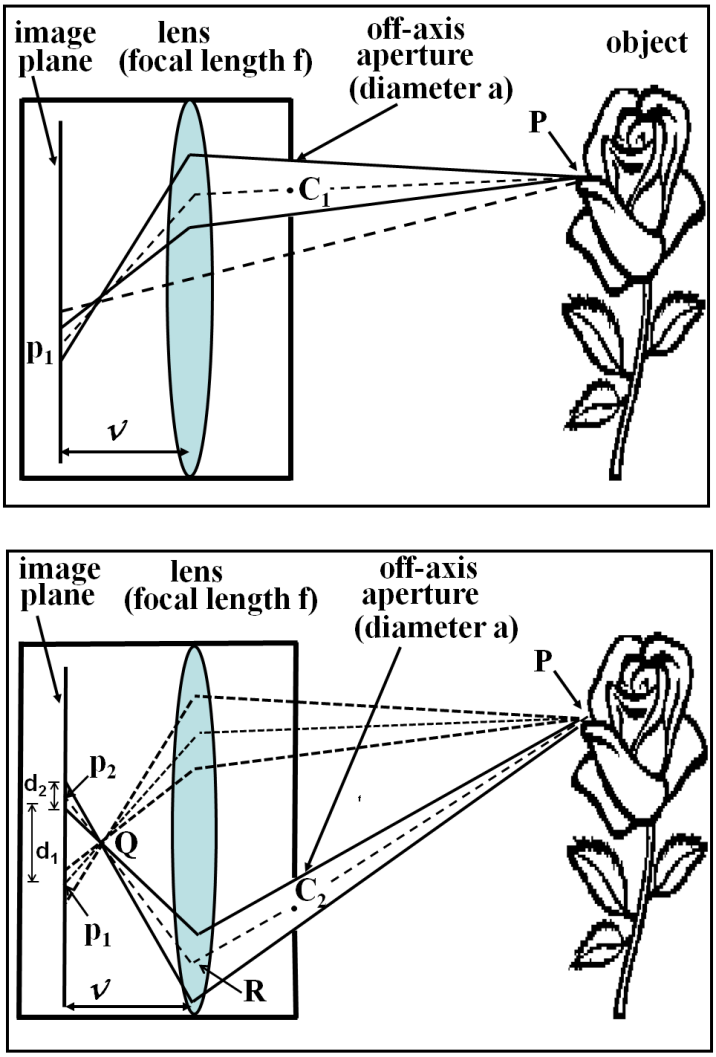


Figure 5.1: Geometry of the off-axis aperture camera. The off-axis aperture camera can be decomposed into three elements: An image plane, a lens, and the moving aperture. Top: Imaging a point **P** on a surface with the aperture positioned at the top. Bottom: Imaging the same point **P** as above with the aperture positioned at the bottom. Notice that when the point being imaged is not in focus, the image is a blurred disc and its center changes with the lens parameters as well as the aperture location. As shown in the lower image, with the location change of the aperture, the center of the blurred disc moves from $\mathbf{p}_1$ to $\mathbf{p}_2$.

5.2.1 The Geometry of the Off-Axis Aperture Camera

The geometry of the off-axis camera is depicted in Figure 5.1. It can be seen by this figure, when a point $\mathbf{P}$ on the object surface is not brought in focus, the image of this point is not a point but a blurred pattern. Because in this work we assume the aperture is a circle, by some explanation in latter part of this section we will know that the shape of the blurred pattern is a disc. From the lower image in Figure 5.1 we can see that with the location change of the aperture the center of the blurred disc will change too. Figure 5.1 depicts an example where the center of the aperture changes from $\mathbf{C}_1$ to $\mathbf{C}_2$ and the center of the blurred disc changes from $\mathbf{p}_1$ to $\mathbf{p}_2$. Figure 5.1 depicts another fact that the diameter d_2 of the blurred disc is proportional to the distance d_1 between $\mathbf{p}_1$ and $\mathbf{p}_2$. This means that, in this camera model, compared with the difference of the locations of the projections of an object the amount of blur is significant. So we combine the stereo and defocus cues in the image formation model proposed for this camera which makes this camera a versatile imaging device. One image of the off-axis camera we used in the experiments is displayed in Figure 5.2.

In this section we will focus on the geometry of this camera model. The image formation model will be given in the next section where we will see how the defocus cue is combined with the stereo cue.

We derive the geometry of this camera based on the lower image displayed in Figure 5.1.

Let the origin of the coordinate system at the center of the lens, z-axis parallel to the optical axis and the x-y plane coincide with the plane of lens. Given that the center of the moving aperture is $\mathbf{C_2} = [C_1\ C_2\ C_3]^T \in \mathbb{R}^3$ and $\mathbf{P} = [P_1\ P_2\ P_3]^T \in \mathbb{R}^3$ is a point on the surface of the object. The ray going through $\mathbf{P}$ and $\mathbf{C}_2$ intersects the lens at point $\mathbf{R}$ and a pencil of rays starting from point $\mathbf{P}$ will converge to point $\mathbf{Q}$ after the diffraction of the lens. Suppose the focal length of the lens f, the distance from the point $\mathbf{Q}$ to the lens v_0 and the distance from the image

Figure 5.2: A image of the off-axis aperture camera.

plane to the lens v. And then the coordinates of the point $\mathbf{Q}$ can be denoted as $[Q_1\ Q_2\ v_0]$. The target is to find the coordinates of the the projection $\mathbf{p}_2$ of point $\mathbf{P}$ on the image plane.

The equation of line $\mathbf{PC}_2$ is:

$$\frac{x - C_1}{P_1 - C_1} = \frac{y - C_2}{P_2 - C_2} = \frac{z - C_3}{P_3 - C_3} \tag{5.1}$$

Put $z = 0$ into Equation (5.1), we can find the first two coordinates of point $\mathbf{R}$ to be

$$R_1 = -C_3\frac{P_1 - C_1}{P_3 - C_3} + C_1, \qquad R_2 = -C_3\frac{P_2 - C_2}{P_3 - C_3} + C_2 \tag{5.2}$$

By the thin lens law, we have $1/f = 1/v_0 + 1/P_3$. So $v_0 = fP_3/(P_3 - f)$. The first two coordinates of point $\mathbf{Q}$ are

$$Q_1 = -\frac{v_0 P_1}{P_3}, \qquad Q_2 = -\frac{v_0 P_2}{P_3} \tag{5.3}$$

The equation of line **RQ** is:

$$\begin{aligned}\frac{P_3x + v_0P_1}{(C_1P_3 - C_3P_1)P_3 + v_0P_1(P_3 - C_3)} &= \frac{P_3y + v_0P_2}{(C_2P_3 - C_3P_2)P_3 + v_0P_2(P_3 - C_3)} \\ &= \frac{z + v_0}{v_0(P_3 - C_3)} \end{aligned} \tag{5.4}$$

Plugging $z = -v$ into Equation (5.4), we obtain the coordinates of image point $\mathbf{p}_2$

$$\begin{aligned} x &= -v\frac{P_1}{P_3} + (1 - \frac{v}{v_0})\frac{C_1P_3 - C_3P_1}{P_3 - C_3} \\ y &= -v\frac{P_2}{P_3} + (1 - \frac{v}{v_0})\frac{C_2P_3 - C_3P_2}{P_3 - C_3} \end{aligned} \tag{5.5}$$

It can be written in a more compact form:

$$\pi[\mathbf{P}] \doteq -v\frac{P_{1,2}}{P_3} + \left(1 - \frac{v}{v_0}\right)\frac{C_{1,2}P_3 - C_3P_{1,2}}{P_3 - C_3} \tag{5.6}$$

where π represents the projection of this camera from $\mathbb{R}^3$ to $\mathbb{R}^2$, $P_{1,2} = [P_1\ P_2]^T$, and $C_{1,2} = [C_1\ C_2]^T$.

Notice that when the image plane is at a distance $v = v_0$ or when the aperture is centered with respect to the lens, i.e., $\mathbf{C} = [0\ 0\ 0]^T$, then the projection π coincides exactly with the perspective projection of $\mathbf{P}$ in a pinhole camera. Otherwise, as the center of the aperture is moved away from $[0\ 0\ 0]^T$, the projection $\pi[\mathbf{P}]$ is a *shifted* perspective projection. Furthermore, when $v \neq v_0$, the point $\mathbf{P}$ generates a blurred pattern on the image plane. In Equation (5.6), with respect to a fixed value C_3, we can see that the coordinates of the projection of an object have linear relationship with $C_{1,2}$ i.e., C_1 and C_2. Since in this work we suppose the aperture is a circle opened on a plane which is parallel to the image plane, then the blurred pattern generated by a surface point on the image plane is also a circle. In this chapter we call it a disc. Let the diameter of the aperture be a, then by computing the projection of $\mathbf{P}$ through the boundary of the off-axis

aperture, we find that the diameter of the blurred disc d_2 as depicted in Figure 5.1 is:

$$d_2 \doteq a \left| 1 - \frac{v}{v_0} \right| \frac{P_3}{P_3 - C_3} \tag{5.7}$$

which is identical to the well-known formula used in shape from defocus when $C_3 = 0$, i.e., when the aperture lies on the lens [33, 53].

5.2.2 Image Formation Model

From the previous section we know that when $v \neq v_0$ the energy from a surface point will distribute within a disc on the image plane. Based on some experiments, we believe that the distribution of the energy within the disc can be described with a Gaussian. we can explicitly write the intensity I measured on the surface of a square pixel $\mathbf{y}_{k,l} \in \Omega$ centered at point (k, l) on the image plane as

$$I(\mathbf{y}_{k,l}) = \int_{k-1/2}^{k+1/2} \int_{l-1/2}^{l+1/2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{e^{-\frac{\|\bar{\mathbf{y}} - \gamma^{-1}\pi[\mathbf{P}(\mathbf{x})]\|^2}{2\sigma^2(\mathbf{x})}}}{2\pi\sigma^2(\mathbf{x})} r(\mathbf{x}) d\mathbf{x} d\bar{\mathbf{y}} \tag{5.8}$$

where σ is related to the blur diameter d_2 via $\sigma \doteq \gamma^{-1}\kappa d_2$, γ is the length of a side of a square pixel. In our experiments we set $\kappa = 1/6$. As for the Gaussian distribution, three standard deviations from the mean account for almost 100% of the total energy. $r(\mathbf{x})$ is the radiance emitted from a point $\mathbf{P}(\mathbf{x})$ on the surface. We parametrized the surface of the object as $\mathbf{P} : \mathbb{R}^2 \mapsto \mathbb{R}^3$. For now we will not make such parametrization explicit as it will be thoroughly analyzed in Section 5.3. Rather, we make our representation of the radiance explicit via the coefficients $r_{i,j}$:

$$r(\mathbf{x}) \doteq \sum_{i,j} r_{i,j} U(\mathbf{x} - \mathbf{x}_{i,j}) \tag{5.9}$$

where $\mathbf{x} = [x_1 \ x_2]^T$, $\mathbf{x}_{ij} = [i \ j]^T$, $U : \mathbb{R}^2 \mapsto \{0, 1\}$ denotes the indicator function, i.e., $U(\mathbf{x})$ is 1 if and only if $-0.5 \leq x_1 < 0.5$ and $-0.5 \leq x_2 < 0.5$. The coordinates (i, j) belong to a regular lattice with step 1. By substituting Equation (5.9) in

Equation (5.8), we obtain

$$\begin{aligned} I(\mathbf{y}_{k,l}) &= \sum_{i,j} r_{i,j} \int_{k-1/2}^{k+1/2} \int_{l-1/2}^{l+1/2} \int_{i-1/2}^{i+1/2} \int_{j-1/2}^{j+1/2} h(\mathbf{x}, \bar{\mathbf{y}}) d\mathbf{x} d\bar{\mathbf{y}} \\ &= \sum_{i,j} r_{i,j} H_{i,j}(\mathbf{y}_{k,l}) \end{aligned} \tag{5.10}$$

where $h(\mathbf{x}, \bar{\mathbf{y}}) \doteq \frac{1}{2\pi\sigma^2(\mathbf{x})} e^{-\frac{\|\bar{\mathbf{y}} - \gamma^{-1}\pi[P(\mathbf{x})]\|^2}{2\sigma^2(\mathbf{x})}}$ denotes the *point spread function* of the camera, and $H_{i,j}(\mathbf{y}_{k,l})$ is implicitly defined by the above equation.

Remark $r_{ij} H_{i,j}(\mathbf{y}_{k,l})$ is the energy given from a unit cell (a small square with side length 1) centered at the regular lattice point (i, j) to a pixel centered at (k, l) on the image plane. There are four integrations involved in the computation of $H_{i,j}(\mathbf{y}_{k,l})$. Computing these integrations directly is time-consuming. By assuming that each small cell is fronto parallel i.e., each small cell is parallel to the image plane, $H_{i,j}(\mathbf{y}_{k,l})$ can be computed in a closed form which is given in the Appendix B.

5.3 Space Warping: How to Bend Geometry to our Advantage

The image formation model in Equation (5.8) could be immediately used to reconstruct the 3D shape of an object in the scene and its radiance. For instance, one could displace the center $\mathbf{C}$ of the aperture on a plane parallel to the image plane and apply standard stereo methods. However, when we only change C_3 or the aperture diameter d_2, the estimation problem becomes more difficult. The main issue is that the relationship between the input images captured with these modalities is highly non linear. To counteract such nonlinearities we suggest changing the representation of the unknowns so that their projection is as linear as possible. We call this method *warping*.

To illustrate the issues created by the original image formation model, we need to make our parametrization of the surface $\mathbf{P}(\mathbf{x})$ explicit. Suppose that $\mathbf{P}(\mathbf{x}) = [\mathbf{x}^T \ u(\mathbf{x})]^T$ where $u : \mathbb{R}^2 \mapsto [0, \infty)$ is the depth map of the scene. Then, using Equation (5.6), a small variation δu of the depth map u causes the projection π to vary as

$$\begin{aligned} \delta\pi[\mathbf{P}(\mathbf{x})] &\doteq \delta u(\mathbf{x}) \tfrac{\partial(\pi[\mathbf{P}(\mathbf{x})])}{\partial u} \\ &= \delta u(\mathbf{x}) \frac{(\mathbf{x} - C_{1,2})(vf + C_3 f - vC_3)}{f(u(\mathbf{x}) - C_3)^2}. \end{aligned} \tag{5.11}$$

One can observe that the variation $\delta\pi[\mathbf{P}(\mathbf{x})]$ in the above equation depends on the coordinates $\mathbf{x}$. In particular, when $\mathbf{x} = C_{12}$, $\delta\pi[\mathbf{P}(\mathbf{x})] = 0$ and as $\|\mathbf{x} - C_{1,2}\|$ grows, also $\|\delta\pi[\mathbf{P}(\mathbf{x})]\|$ grows. While in principle this behavior is acceptable, in practice it is an issue when using gradient-based techniques; part of the gradient of the cost functional will depend on $\delta\pi[\mathbf{P}(\mathbf{x})]$ and, as a consequence, the points close to the point $\mathbf{x} = C_{12}$ will go much more slowly than the points far away from the point $\mathbf{x} = C_{12}$. If enlarge the time step, the convergence of the points far away from the point $\mathbf{x} = C_{12}$ will be unstable.

We suggest a simple method to offset this behavior. The key idea is to choose a parametrization of the surface of the object $\mathbf{P}(\mathbf{x})$ such that the projection $\pi[\mathbf{P}(\mathbf{x})] = \alpha\mathbf{x}$ for some scalar $\alpha \neq 0$, and $\mathbf{P}(\mathbf{x})$ does not depend on the varying camera parameters. If such a projection exists, then the captured images can be easily warped into each other so that the only difference between them is the relative amount of defocus. Once the (warped) depth map u is recovered from the warped images, one has to undo the warping by using the parametrization $\mathbf{P}(\mathbf{x})$ (unwarping). For instance, in shape from defocus where $\mathbf{C} = [0\ 0\ 0]^T$ and v changes between the input images. In this situation, plug $C_1 = C_2 = C_3 = 0$ into Equation (5.11), one has that $\delta\pi[\mathbf{P}(\mathbf{x})] = \delta u(\mathbf{x}) \frac{v\mathbf{x}}{u^2(\mathbf{x})}$, which is still dependent on $\mathbf{x}$, we cannot use the parametrization $\mathbf{P}(\mathbf{x}) = [\mathbf{x}^T \ u(\mathbf{x})]^T$. Let us define $\mathbf{P}(\mathbf{x}) \doteq [\mathbf{x}^T \ 1]^T u(\mathbf{x})$. Then, the projection $\pi[\mathbf{P}(\mathbf{x})] = -v\mathbf{x}$ and $\delta\pi[\mathbf{P}(\mathbf{x})] = 0$; the input images can be mapped into each other by warping their image domain (i.e.,

scaling each image by its corresponding v and γ)

$$\begin{aligned}\hat{I}(\mathbf{y}_{k,l}, v) &\doteq \gamma^{-2} v^2 I(\gamma^{-1} v \mathbf{y}_{k,l}) \\ &= \gamma^{-2} v^2 \sum_{i,j} r_{i,j} \int_{k-1/2}^{k+1/2} \int_{l-1/2}^{l+1/2} \int_{i-1/2}^{i+1/2} \int_{j-1/2}^{j+1/2} \frac{e^{-\frac{\|\gamma^{-1} v \bar{\mathbf{y}} - \gamma^{-1} v \mathbf{x}\|^2}{2\sigma^2(\mathbf{x})}}}{2\pi\sigma^2(\mathbf{x})} d\mathbf{x} d\bar{\mathbf{y}} \\ &= \sum_{i,j} r_{i,j} \int_{k-1/2}^{k+1/2} \int_{l-1/2}^{l+1/2} \int_{i-1/2}^{i+1/2} \int_{j-1/2}^{j+1/2} \frac{1}{2\pi\hat{\sigma}^2(\hat{\mathbf{x}})} e^{-\frac{\|\bar{\mathbf{y}} - \mathbf{x}\|^2}{2\hat{\sigma}^2(\mathbf{x})}} d\mathbf{x} d\bar{\mathbf{y}}\end{aligned} \tag{5.12}$$

where $\hat{\sigma}(\mathbf{x}) = \gamma\ v^{-1}\ \sigma(\mathbf{x})$.

Finally, once $u(\mathbf{x})$ is reconstructed, one needs to apply the parametrization $[\mathbf{x}^T\ 1]u(\mathbf{x})$ to obtain the correct depth map.

Remark Most algorithms for shape from defocus use implicitly this warping of the depth map. In fact, typically one works with images that have been aligned (warped) and the projection $\pi[\mathbf{P}(\mathbf{x})]$ is always approximated by $\mathbf{x}$. However, in most methods the last step (unwarping) is not applied, thus preventing the algorithms from reconstructing the correct object. It is interesting to note that in the case of telecentric optics [181] some further simplifications are possible. We have that $\mathbf{C} = [0\ 0\ f]$, and hence $\pi[\mathbf{P}] \doteq -\frac{f\mathbf{x}}{u(\mathbf{x})-f}$ does not change between the images as it does not depend on v. With telecentric optics a warping between the input images is not required (as the projections do not change with v). However, as the projection is still a function of the depth map u and $\mathbf{x}$, the parametrization still needs to be changed (e.g., by setting $\mathbf{P}(\mathbf{x}) \doteq [\mathbf{x}^T\ 1]^T(u(\mathbf{x}) - f)$) and the corresponding reconstruction still needs to be unwarped.

In the case of varying the diameter of the aperture a, the analysis is fairly straightforward as the projection π does not depend on a. Hence, we can choose $\mathbf{P} \doteq [P_{1,2}\ P_3]^T$ where

$$\mathbf{P}_{1,2}(\mathbf{x}) \doteq \frac{f(u(\mathbf{x}) - C_3)\mathbf{x} - (fu(\mathbf{x}) - vu(\mathbf{x}) + vf)C_{1,2}}{C_3 v - C_3 f - vf} \tag{5.13}$$

and $P_3 \doteq u(\mathbf{x})$. It is immediate to see that for such a parametrization the projection $\pi[\mathbf{P}(\mathbf{x})] = \mathbf{x}$. As in the case of telecentric optics, no warping of the input images is required.

The case of varying C_3 is instead more difficult. We approximate the distance of the aperture from the lens C_3 with an average value $\bar{C}_3$. Then, let $\mathbf{P}(\mathbf{x}) = [\mathbf{x}^T \ \frac{u(\mathbf{x})}{u(\mathbf{x})-\bar{C}_3}]^T(u(\mathbf{x}) - \bar{C}_3)$ so that

$$\pi[\mathbf{P}(\mathbf{x})] \simeq -\frac{vf + (f-v)C_3}{f}\mathbf{x} + \left(1 - \frac{v}{v_0(\mathbf{x})}\right)\frac{u(\mathbf{x})C_{1,2}}{u(\mathbf{x}) - C_3}. \tag{5.14}$$

The variation $\delta\pi[\mathbf{P}(\mathbf{x})] \simeq \frac{\delta u(\mathbf{x})C_{1,2}}{f(u(\mathbf{x})-C_3)^2}(C_3(v-f) - vf)$ is then approximately independent of $\mathbf{x}$ (although it remains dependent on $u(\mathbf{x})$) and the input images can be pseudo-aligned simply by scaling the image domain by $\frac{vf+(f-v)C_3}{f}$. After the pseudo-alignment, the resulting projection is then a function

$$\pi[\mathbf{P}(\mathbf{x})] \simeq -\mathbf{x} + \frac{1 - u(\mathbf{x})\left(\frac{1}{f} - \frac{1}{v}\right)}{1 - C_3\left(\frac{1}{f} - \frac{1}{v}\right)}\frac{C_{1,2}}{u(\mathbf{x}) - C_3}. \tag{5.15}$$

Remark Notice that in general it is not straightforward to find a parametrization $\mathbf{P}$ so that the projection does not depend on the depth map u. In such case we can approximately offset some of the undesired behavior due to the variation of the projection with respect to variations of the depth map.

5.4 A Gradient Flow Algorithm

To find the unknown depth map u and the radiance r, we pose the following minimization problem

$$\begin{aligned} \hat{u}, \hat{r} &= \arg\min_{u,r} E \\ &= \arg\min_{u,r} \sum_{n=1}^{N} \sum_{k,l} \left(I_n(\mathbf{y}_{k,l}) - J_n(\mathbf{y}_{k,l})\right)^2 \\ &\quad + \lambda_1 \int \|\nabla u(\mathbf{x})\|^2 d\mathbf{x} + \lambda_2 \int \|r(\mathbf{x}) - r^*(\mathbf{x})\|^2 d\mathbf{x} \end{aligned} \tag{5.16}$$

where r^* is a reference radiance (e.g., one of the input images or an average of the input images), and λ_1, λ_2 are positive scalars that control the amount of regularization. n is the index of the images and E is implicitly defined by Equation (5.16). While the first term on the right hand side of Equation (5.16) matches the image model I_n to the measured images J_n for varying aperture center $\mathbf{C}$ and diameter a, the second term introduces a smoothness constraint on the recovered depth map, and the third term prevents the estimated radiance from growing unboundedly.

The minimization is carried out by the following gradient flow algorithm. First we compute the associated Euler-Lagrange equations. Let the Euler-Lagrange equations of Equation (5.16) with respect to u and r be denoted by $\mathcal{L}_u E$ and $\mathcal{L}_r E$, they have the following forms:

$$\begin{aligned} \mathcal{L}_u E(\mathbf{x}_{i,j}) &= 2 \sum_{n=1}^{N} \sum_{k,l} \Delta I_n(\mathbf{y}_{k,l}) \frac{\partial I_n(\mathbf{y}_{k,l}, \mathbf{x}_{i,j})}{\partial u} - 2\lambda_1 \Upsilon(u(\mathbf{x}_{i,j})) \\ \mathcal{L}_r E(\mathbf{x}_{i,j}) &= 2 \sum_{n=1}^{N} \sum_{k,l} \Delta I_n(\mathbf{y}_{k,l}) H^n_{i,j}(\mathbf{y}_{k,l}) + 2\lambda_2 (r(\mathbf{x}_{i,j}) - r^*(\mathbf{x}_{i,j})) \end{aligned} \tag{5.17}$$

where $H^n_{i,j}(\mathbf{y}_{k,l})$ has been defined in Equation (5.10) and $\Delta I_n(\mathbf{y}_{k,l}) \doteq (I_n(\mathbf{y}_{k,l}) - J_n(\mathbf{y}_{k,l}))$

and

$$\frac{I_n(\mathbf{y}_{k,l}, \mathbf{x}_{i,j})}{\partial u} \doteq \int_{k-1/2}^{k+1/2} \int_{l-1/2}^{l+1/2} \int_{i-1/2}^{i+1/2} \int_{j-1/2}^{j+1/2} h_n(\mathbf{x}, \bar{\mathbf{y}}) \frac{\partial h_n(\mathbf{x}, \bar{\mathbf{y}})}{\partial u} r(\mathbf{x}) d\mathbf{x} d\bar{\mathbf{y}} \quad (5.18)$$

where

$$\begin{aligned} \frac{\partial h_n(\mathbf{x}, \bar{\mathbf{y}})}{\partial u} &\doteq \frac{(\bar{\mathbf{y}} - \gamma^{-1}\pi[\mathbf{P}(\mathbf{x})])}{\sigma^2(\mathbf{x})} \gamma^{-1} \frac{\partial \pi[\mathbf{P}(\mathbf{x})]}{\partial u}) \\ &+ \left(\frac{\|\bar{\mathbf{y}} - \gamma^{-1}\pi[\mathbf{P}(\mathbf{x})]\|^2}{\sigma^3(\mathbf{x})} - \frac{1}{\sigma(\mathbf{x})} \right) \frac{\partial \sigma(\mathbf{x})}{\partial u} \end{aligned} \quad (5.19)$$

and

$$\frac{\partial \sigma(\mathbf{x})}{\partial u} = \gamma^{-1} \kappa a \, \text{sign} \left(1 - \frac{v}{v_0(\mathbf{x})} \right) \frac{vC_3 - vf - C_3 f}{f(u(\mathbf{x}) - C_3)^2} \quad (5.20)$$

and

$$\Upsilon(u(\mathbf{x})) = \frac{\partial u}{\partial x_1} \frac{\partial u^2}{\partial^2 x_1} + \frac{\partial u}{\partial x_2} \frac{\partial u^2}{\partial^2 x_2} \quad (5.21)$$

where $\mathbf{x} = [x_1 \; x_2]^T$ and $\frac{\partial u^2}{\partial^2 x}$ denotes the second derivative of u with respect to x.

As for the term $\frac{\partial \pi[\mathbf{P}(\mathbf{x})]}{\partial u}$, after the space warping, this term will be simplified and when $\pi[\mathbf{P}(\mathbf{x})]$ does not depend on the depth u, this term is zero.

By parameterizing the descent direction by an artificial time $t \geq 0$, the gradient flow can be easily modified into more efficient schemes without affecting the location of the minima. The minimization can be carried out by the following gradient flow algorithm

$$\frac{\partial u(\mathbf{x}, t)}{\partial t} = -\mathcal{L}_u E \qquad \frac{\partial r(\mathbf{x}, t)}{\partial t} = -\mathcal{L}_r E \quad (5.22)$$

where the derivative with respect to iteration time is approximated by a forward difference, and $\mathcal{L}_u E$ and $\mathcal{L}_r E$ are given in Equation (5.17).

As can be seen in most formulas, this algorithm can be implemented very efficiently. One can tabulate most computations from analytic solutions of the integrals of Gaussian over finite domains (i.e., via the error function) and by exploiting the separability of the Gaussian function. Indeed our current implementation in C++ with a Pentium $2GHz$ takes about 1 second to compute the

above gradients on an image of dimensions 200×200 pixels. Each experiment displayed in this chapter can be finished within 200 iterations of the shape reconstruction and image restoration. So the total time taken by any experiment presented in this chapter is less than 3 minutes.

5.5 Experiments

5.5.1 Experimental Results from Synthetic Data

In this section, we test the proposed algorithm on four synthetically-generated data: An equifocal plane, a scene made of equifocal planes at different depths (`Cube` dataset), a slope (`Slope` dataset), and a wave (`Wave` dataset). For each of these shapes we show the reconstruction results obtained by either changing the diameter a of the aperture (see Figure 5.3 and 5.4) or the distance C_3 between the aperture and the lens (see Figure 5.5 and 5.6).

In the experiments displayed in Figure 5.3 and 5.4, we use 4 input images, which are obtained by setting the aperture diameter a as $2mm$, $4mm$, $5mm$, and $6mm$ and with other parameters fixed to focal length $f = 30mm$, $\mathbf{C} = [0\ 0\ 20mm]^T$, and distance image plane to lens $v = 45mm$. In Figure 5.3 we show only two of the input images corresponding to $a = 2mm$ and $a = 6mm$. In the same figure we show the true radiance and the estimated radiance. In Figure 5.4 we show the ground truth for the shapes of the `Cube`, `Slope`, and `Wave` datasets in the left column, the corresponding reconstructed shapes from the input images without noise in the middle column and the corresponding reconstructed shapes from the input images along with 2% added Gaussian noise in the right column.

In the experiments displayed in Figure 5.5 and 5.6, 3 input images are obtained by setting the distance of the aperture to the lens C_3 as $3mm$, $8mm$, and $12mm$ and with other parameters fixed to focal length $f = 30mm$, $a = 5mm$, and distance image plane to lens $v = 45mm$. The image order arranged in Figure 5.5

and 5.6 are the same as in Figure 5.3 and 5.4 respectively. The two input images displayed in Figure 5.5 are generated with $C_3 = 3mm$ and $C_3 = 12mm$.

In all experiments (including the ones on real data) the shape is initialized as a plane parallel to the image plane and the radiance is initialized as one of the input images.

To show the robustness of the method, we test our algorithm with the images with additive Gaussian noise at 5 different levels. They are generated by adding 1%, 2%, 5%, 15% and 20% of Gaussian noise of the radiance intensity to the input images. In Figure 5.7 we show the absolute errors and relative errors of the experiments with each dataset at different noise level. which we compute as the average of the differences of L_2 norm between the estimated shape and the true shape, and the relative error, which we compute as the absolute error and then normalized by dividing the average of the L_2 norm of the true shape. The errors in the case of changing the aperture diameter are shown in the top row in Figure 5.7 and the errors in the case of changing the distance from the aperture to the lens are shown in the bottom row in Figure 5.7.

As we can see from the Figure 5.7, in both group of experiments, as the noise level going larger, the error in each case is also going higher. They have a approximately linear relationship. Compared among different datasets, the worst case is always coming from the `Wave` dataset and the best case is always from the equifocal plane dataset, it seems that the sharper of the oscillations on the object surface the worse is the reconstruction shape. There are at least two possible reasons. One is that in our image formation model, after the discretization of the object surface, we assume that each small cell of the surface is fronto parallel. A surface with sharp oscillations will violate this assumption. Another reason is the smoothing term in the formulation. The smoothing term smear out the surface part with steep gradient. This affects the accuracy of the reconstruction of the `Wave` dataset. Although the `Cube` dataset is piecewise fronto parallel, because the effects of the smoothing term the results from the `Cube` dataset are worse than

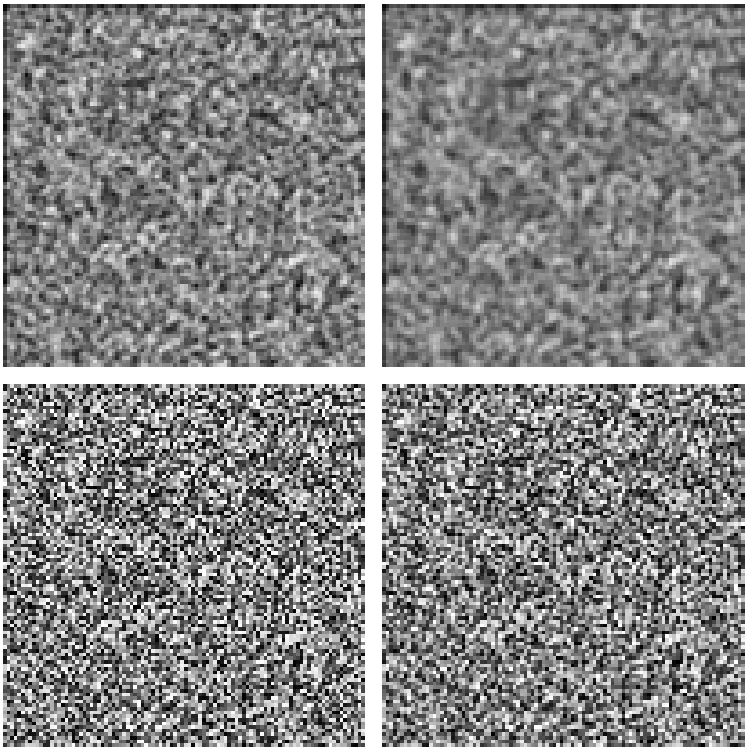


Figure 5.3: Image restoration when changing the size of the aperture. First row shows two of the four input images. The first image in the second row is the true radiance and the second one is the restored radiance.

those from the slope dataset.

5.5.2 Experimental Results from Real Data

To illustrate the algorithm above and validate it empirically, we test it on real images. Here we show an experiment of a scene containing a miniature house. The distance between the camera and the house model is about 400 millimeters. The images are captured by a Nikon D80 Digital Camera equipped with a Nikon AF Nikkor lens with $F = 50mm$, $v = 57.57mm$, and $\gamma = 23.7pixels/mm$. The four input images are captured by setting the aperture to $5mm$, $9mm$, $15mm$ and $22mm$. Due to the difference in the aperture, the intensities of the captured images need to be corrected. We found the average intensity of each image. By choosing a value as the criterion, we compute the ratios of the average intensities

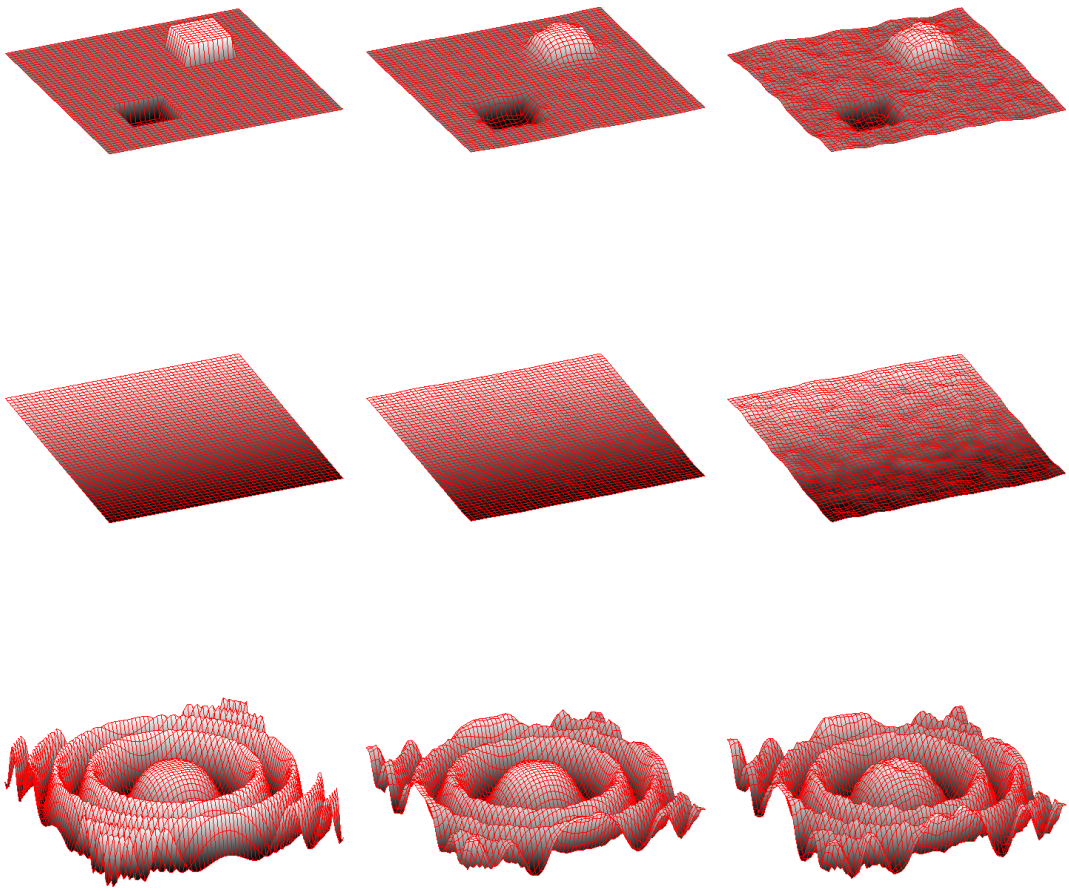


Figure 5.4: The reconstructions of the `Cube`, `Slope` and `Wave` datasets when changing the size of the aperture. In each row, the first image displays the true shape of the corresponding dataset, the second and third images display the reconstructed shapes without noise and with 2% noise.

against this value. The final images are obtained from the original images divided by the corresponding ratios. Two of the input images are shown on the top row of Figure 5.8. On the bottom row we show the restored radiance (left) and the recovered shape (right). Figure 5.9 shows several views of the reconstructed scene. As one can see the qualitative shape has been successfully captured.

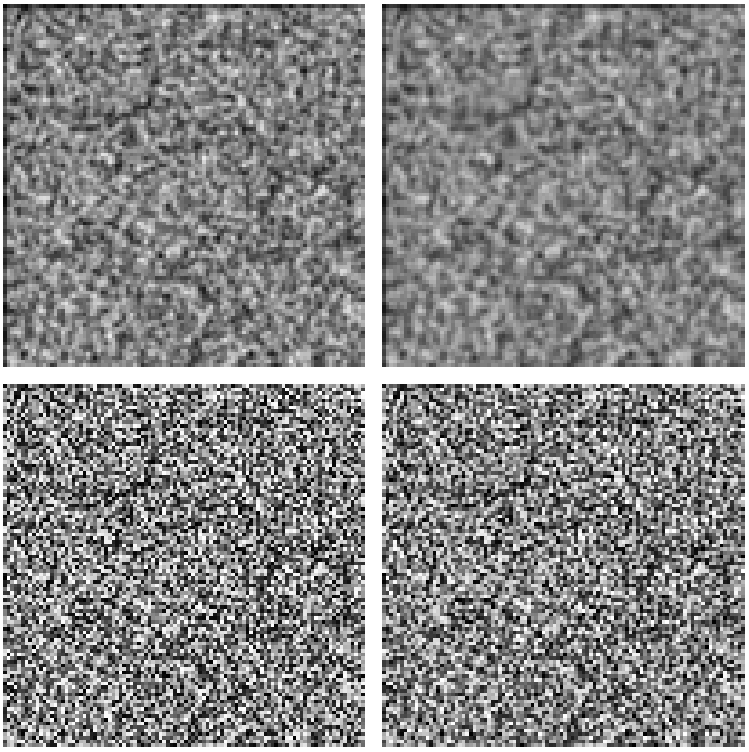


Figure 5.5: Image restoration by changing the distance of the aperture from the lens. First row shows two of the four input images. The first image in the second row is the true radiance and the second one is the restored radiance.

5.6 Conclusions

We presented a novel approach to shape estimation and image restoration algorithms based on the off-axis aperture camera. The main property of this camera is that it needs neither a relative motion with respect to the object nor a change in the image plane and lens position. The only moving part is the aperture. We change its location and its diameter and show how this relates to the 3-D structure of the scene via the image formation model. We propose several parameterizations for the geometry of the object so that the resulting minimization algorithm is simple, and its convergence well-behaved. Because this is the first time that a camera is modeled in this way, so there is no comparison possible. But the results on several synthetic and real datasets are shown and demonstrate

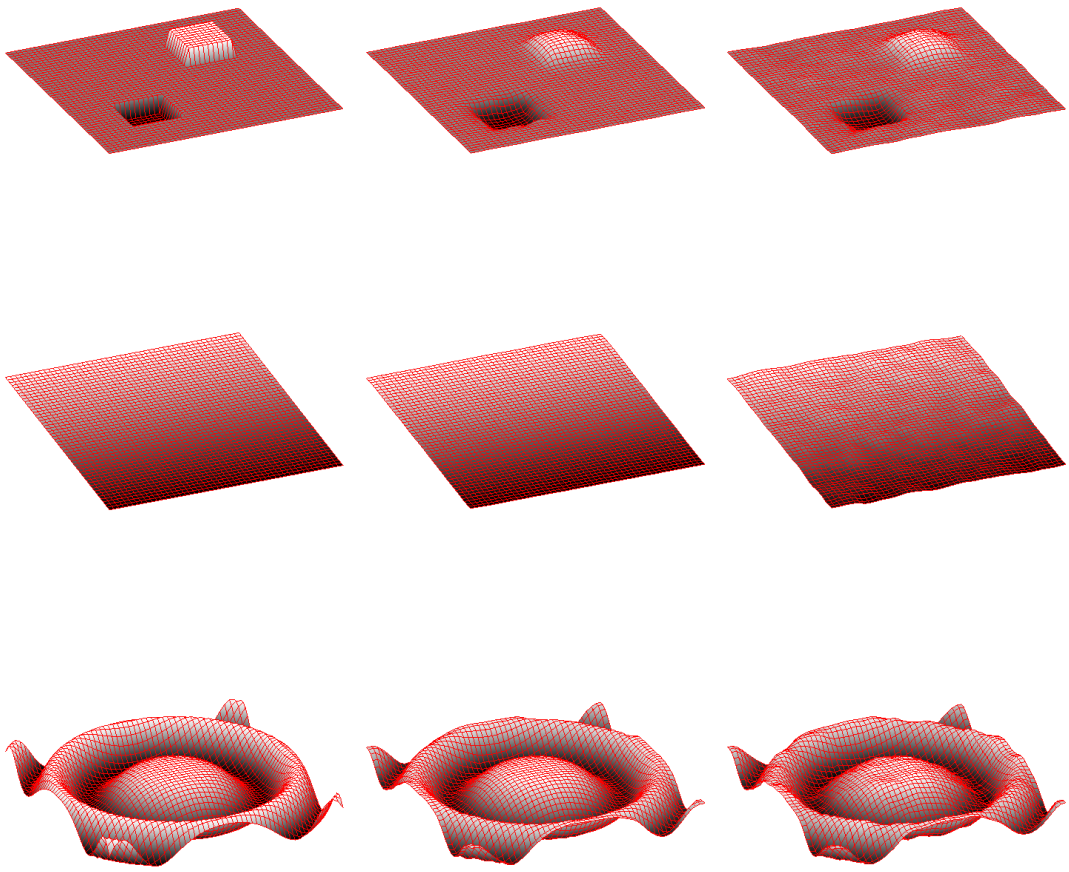


Figure 5.6: Shapes reconstructed when changing the distance from the aperture to the lens. In each row, the first image displays the true shape of the corresponding dataset, the second and third images display the reconstructed shapes without noise and with 2% noise.

the accuracy and effectiveness of the technique. From the experimental results we can see that this approach works better for smooth surfaces. For the surfaces with sharp oscillations, some parts of the reconstructed surface have been killed or smoothed out in some way. There are two possible reasons for this. One is the assumption in the image formation model where we assume that each small cell on the discretized object surface is fronto parallel. The other is the smoothing terms in the formulation. The work of expanding the image formation model to a more general one and the work of studying the effects of the smoothing terms

will be done in the future.

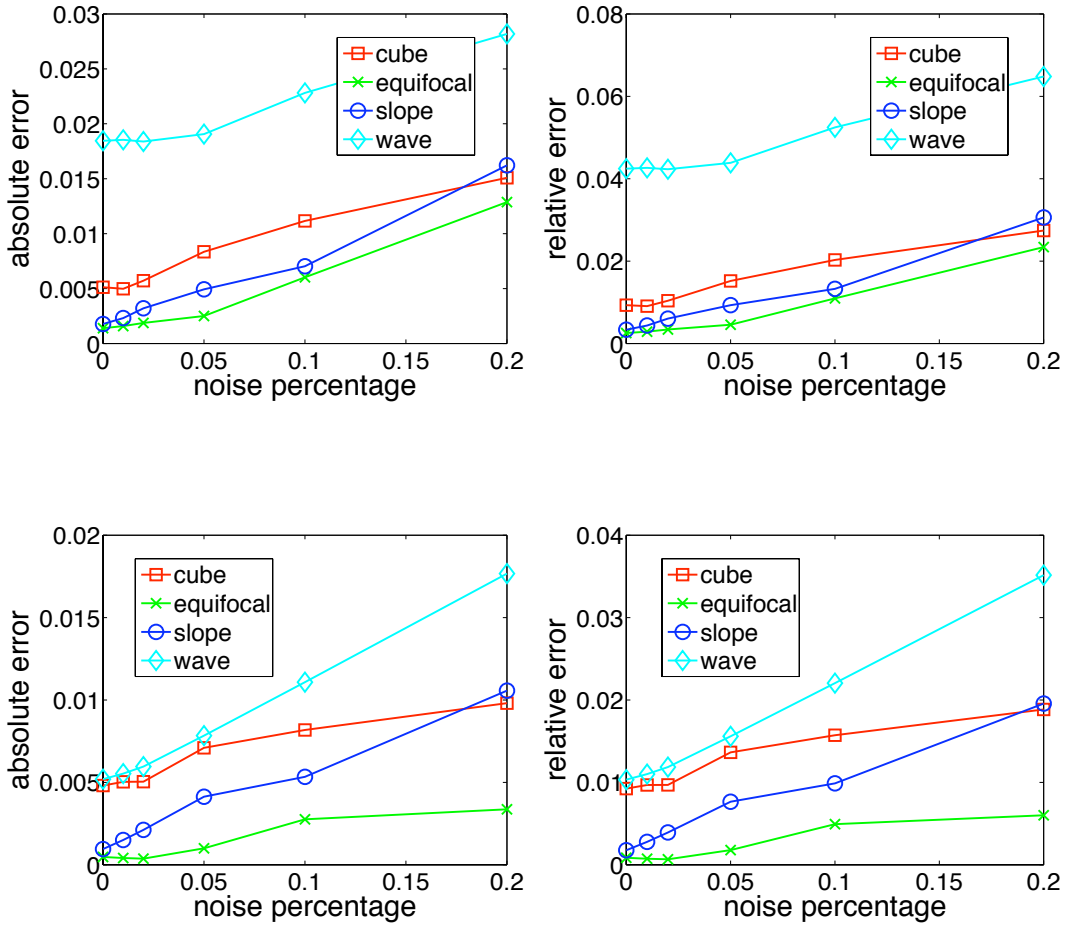


Figure 5.7: Absolute and relative errors between the ground truths and the estimated depth maps of the four synthetic datasets at six noise levels. The unit of the absolute error is millimeter. Top row: the results from the experiments when changing the size of the aperture. Bottom row: the results from the experiments when changing the distance from the aperture to the lens.

Figure 5.8: Experimental results on real data. Top row: two of the input images; the image on the left has been captured with $A = 5mm$, while the image on the right has been captured with $A = 22mm$. Second row: the restored radiance(left) and reconstructed depth map in gray scale (right).

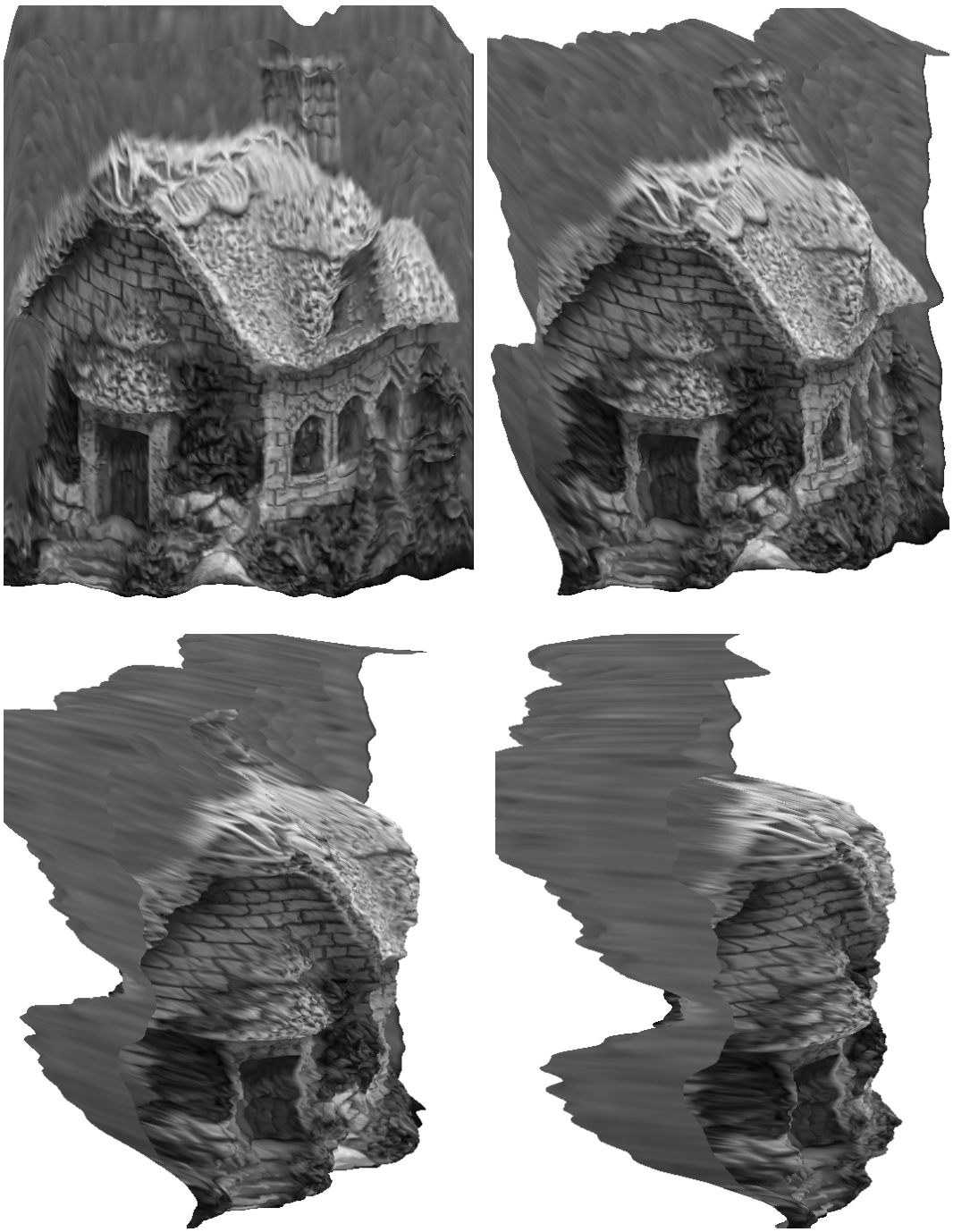


Figure 5.9: A few views of the reconstructed house displayed in Figure 5.8.

Chapter 6

Large-Baseline Multiview Stereo: Dealing with View-varying Clutter and Occlusions

6.1 Introduction and Problem Statement

In this chapter we present two novel methods for solving calibrated multiview stereo (MVS), i.e., the problem of estimating 3D surfaces from a collection of 2D views with known pose. Part of the work presented in this chapter is published in our paper [44].

Research in MVS is very active in the field of computer vision and a wide variety of methods have been proposed to address the recovery of the surface of 3D objects in very challenging scenarios, e.g., for wide-baseline datasets [80], with non-Lambertian objects [20, 171], dealing with illumination [86], or in the presence of clutter [62].

We pose the problem of estimating the surface of Lambertian objects in the scene from multiple calibrated views as the problem of determining whether a point in space (a voxel) lies inside or outside any of the objects. The estimated

surface is then implicitly defined as the interface separating the two groups of voxels. As pointed out by Furukawa and Ponce [62], MVS algorithms can be associated to the datasets that they can handle: objects with clear silhouettes; objects in clutter; objects with occlusions. When the silhouettes of the objects in images are available, the silhouette images can be used to generate the visual hull directly. Either the visual hull can be regard as the final reconstruction or it can be used as an initialization for the final reconstruction. To finish the 3D reconstruction based on visual hull makes life easier. Unfortunately, if the images are not taken in a controlled environment, it is not easy or even possible to extract the silhouettes. Our methods can deal with all the three above scenarios and focus on the challenge posed by view-varying occlusions and clutter. We propose general solutions that do not require knowledge of the visual hull, silhouettes or approximate depth maps either implicitly or explicitly.

In the first part of this chapter, we introduce our first method that can deal with all the three above scenarios in a globally optimal fashion. The experimental results with this method with the `two_tori` datasets with or without view-varying occlusions are depicted in Figure 6.1. As we can see, the texture of the background is similar to that of the tori, so that there is no clear boundary between the foreground and the background. We call this type of background clutter. Also the background is changing from view to view so the clutter is view-varying. As it has been pointed out in Chapter 1, the occlusions we meet in 3D reconstruction can be divided into two classes: self-occlusions and occlusions from other objects. From the `two_tori` datasets (Figure 6.1) we can see that some parts of the tori are self-occluded. In each of the two images displayed in the second row, there is a disc occluding some parts of the tori. These discs do not belong to the objects we want to reconstruct, so they are occlusions from other objects. Because the discs are located randomly in each view, they appear at different locations relative to the tori in different views. We call them *view-varying* occlusions.

Our method is a continuous convex formulation of MVS (Section 6.2) which

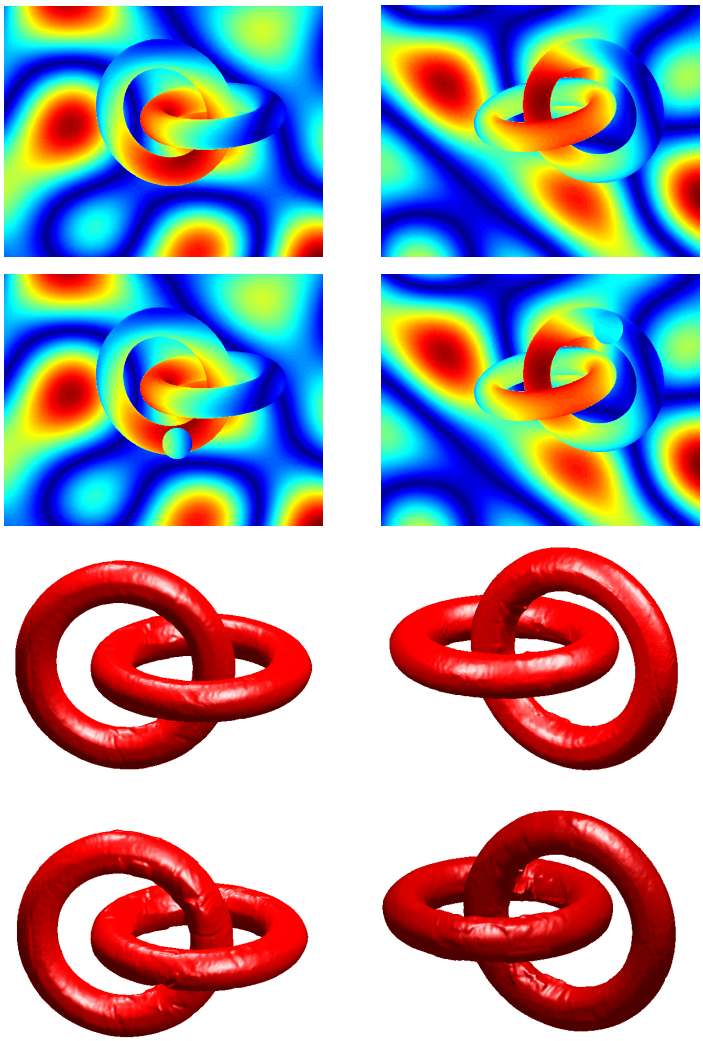


Figure 6.1: Reconstruction of two concatenated tori from synthetic images where the foreground and the background have the same texture (clutter) and with or without view-varying occlusions (circular regions that change position in each frame). Top row: two of the input images of the `two_tori` dataset with only view-varying clutter. Second row: two of the input images of the `two_tori` dataset with view-varying clutter and occlusions. Third row: two views of the reconstructed 3D model from the `two_tori` dataset with only view-varying clutter. Bottom row: two views of the reconstructed 3D model from the `two_tori` dataset with view-varying clutter and occlusions.

includes a Bayesian formulation of the visibility of each camera (Section 6.3). In Section 6.4 we explicitly deal with the integration of 3D surface estimates from different views. The convex cost functional is then minimized by an efficient gradient-flow algorithm in Section 6.5. Finally, in Section 6.6 we demonstrate the method on data with view-varying clutter and occlusions and on the original Middlebury data set, where we show that, with the clutter and occlusions, it still performs similarly to current state-of-the-art methods.

The contributions of this method are that we formulate MVS to simultaneously deal with the dependency of multiple views by voting the visibility functions with the proposed `Robust interpolating` function. The proposed algorithm is independent of initialization, can easily incorporate surface regularization terms, does not have degenerate solutions (e.g., the empty set), and does not use the visual hull or silhouettes explicitly or implicitly at any step of the algorithm. Furthermore, more general models of image formation can be used without compromising the estimation process.

In the second part of this chapter, we introduce our second method for solving the same problem. In this method, a narrow band around the approximated surface is chosen for each central view. For a voxel close to the surface, the number of narrow bands containing this voxel will be counted. If a certain narrow band contains a voxel, we would say the corresponding central view has a contribution to this voxel. Then how many views contribute to a certain voxel is determined by the number of the narrow bands which contain this voxel. The final visibility value of this voxel will be determined by all the views which contribute to this voxel. By partitioning the voxels within the narrow band into two groups according to their visibility values, the surface is reconstructed as the interface between these two groups. The benefits of doing this are the follows. First, this method has the same robustness to view-varying occlusions and clutter as the preceding method. Second, the computational load in this method has been reduced dramatically by not using the interpolating function to integrate the visibility functions from

each view into a global visibility function. A shortcoming of this method is that the proposed algorithm is not convex. But as shown by the experimental results, by setting some parameters properly, it seems that the convergence to the global minimum is not a problem.

The proposed algorithm is based on the work of Chan and Vese [31], whose algorithm is used for segmentations of 2D images. It is also related to the work of Curless and Levoy [40] where a signed distance function is generated based on the surface from the range image, and then the narrow bands of weighted signed distance function are accumulated to generate a full surface. Curless' method works well with range images generated by active sensors where there is less noise than the depth maps generated from 2D images. Goesele et al. [65] employ this method to work on the depth maps generated from 2D images, but only the portion of the scene that can be matched with high confidence in each input view can be reconstructed and leading to that each individual depth map may contain numerous holes.

The rest of this chapter is organized as follows: first the energy formulation is described in Section 6.7 which is followed by the description of the extraction of narrow bands in Section 6.8. We introduce our techniques to extract the surfaces in Section 6.9. After the experimental results are given in Section 6.10, this chapter is terminated by a brief conclusion on both methods.

6.2 A Continuous and Convex Formulation of Multiview Stereo

We pose the problem of estimating the surface of Lambertian objects in the scene from multiple calibrated views as the problem of determining whether a point in space (a voxel) lies inside or outside any of the objects. This is represented with a function $\phi : \mathbf{V} \subset \mathbb{R}^3 \mapsto -1, 1$, with $\mathbf{V}$ the bounded volume in 3D space where

reconstruction is performed. Ideally, in our approach when a voxel $\mathbf{X} \in \mathbf{V}$ is inside any of the objects in the scene then the value of function ϕ at this voxel is 1 and if a voxel is outside any of the objects in the scene then the value of function ϕ at that voxel is -1. The function ϕ is called *visibility* function in this chapter. The surface of the objects is defined implicitly as the set of points $\{\mathbf{X} : \phi(\mathbf{X}) = 0\}$. The next step is to define an energy such that its minimum is at the surface of the objects in the scene. To do so we introduce the following energy minimization

$$\begin{aligned}\hat{\phi} &= \arg\min_{\phi} E[\phi] \\ &\doteq \int \Phi(\tilde{\phi}(\mathbf{X})|\phi(\mathbf{X}))d\mathbf{X} + \alpha \int \Psi(\tilde{\phi}(\mathbf{X}))|\nabla\phi(\mathbf{X})|d\mathbf{X} \\ &\quad +\beta \int \theta(\phi(\mathbf{X}))d\mathbf{X}. \end{aligned} \tag{6.1}$$

The energy is composed of three terms. The first term $\Phi(\tilde{\phi}|\phi)$ is the *data fidelity* term, which measures the discrepancy between $\tilde{\phi}$ and ϕ. In our notation the function $\tilde{\phi}$ is an approximate estimate of the visibility function which is computed from the input data. The details of the computation of the visibility function are given in the following sections. The discrepancy function will be analyzed in the next section. The second term is the regularization term. It is a weighted version of Total Variation, which penalizes large variations of ϕ at the surface of the object. The weight is defined as $\Psi(\tilde{\phi}) \doteq \exp[-\tilde{\phi}^2/\mu]$ with a positive constant μ. The second term is weighted by a positive constant α. The third term is a convex penalty term with $\theta(\epsilon) \doteq \max\{0, |\phi| - 1\}$ and weighted by a positive constant β. This term prevents ϕ from leaving the range $[-1, 1]$ and keep this formulation as a convex one.

6.2.1 The Discrepancy Functions in the Formulation

In our implementation we tested two choices for Φ. One choice is the discrepancy

$$\Phi(\tilde{\phi}(\mathbf{X})|\phi(\mathbf{X})) = |\tilde{\phi}(\mathbf{X}) - \phi(\mathbf{X})| \tag{6.2}$$

and a second choice is

$$\Phi(\tilde{\phi}(\mathbf{X})|\phi(\mathbf{X})) = (1 + \tilde{\phi}(\mathbf{X}))(1 - \phi(\mathbf{X})) + (1 - \tilde{\phi}(\mathbf{X}))(1 + \phi(\mathbf{X})). \tag{6.3}$$

If we assume that $-1 \leq \phi \leq 1$, the relationship between $\Phi(\tilde{\phi}|\phi)$, $\tilde{\phi}$ and ϕ expressed by Equation (6.3) is illustrated in Figure 6.2. Given two constant values $\tilde{\phi}_0$ and $\tilde{\phi}_1$ with $-1 \leq \tilde{\phi}_0 < 0$ and $0 < \tilde{\phi}_1 \leq 1$, it can be seen that Φ obtains its minimum at $\phi = -1$ and $\phi = 1$ along $\tilde{\phi} = \tilde{\phi}_0$ and $\tilde{\phi} = \tilde{\phi}_1$ respectively. In fact Φ obtains its minimum at $\phi = 1$ for all $\tilde{\phi} \in (0, 1]$ and minimum at $\phi = -1$ for all $\tilde{\phi} \in [-1, 0)$. It means that for all voxels with $-1 \leq \tilde{\phi} < 0$ the value of ϕ will converge to -1 to minimize Φ and for all voxels with $0 < \tilde{\phi} \leq 1$ the value of ϕ will converge to 1 to minimize Φ. The only uncertain situation is when $\tilde{\phi} = 0$ where Φ takes a constant value. In this situation, ϕ can be any value between -1 and 1, but in practice, the regularization term in the formulation will do its work to pick a suitable value for ϕ. In fact, if $\tilde{\phi}$ is either -1 or $+1$ for most of the voxels and has a quick transition through 0 at the surface of the objects, then we have found no noticeable difference between the solution obtained with Equation (6.2) and the solution obtained with Equation (6.3) in our experiments.

The interpretation of the term $\tilde{\phi}$ in Equation (6.2) is much more apparent than in Equation (6.3): It behaves as a *proxy*, i.e., as an initial estimate of the function ϕ obtained from the data. Then, by minimizing Equation (6.1) we approximate the proxy with a smooth function. An immediate consequence of this formulation is that the accuracy of the solution depends highly on the accuracy of the proxy. In the following sections, we will study how to calculate $\tilde{\phi}(\mathbf{X})$ so that we can

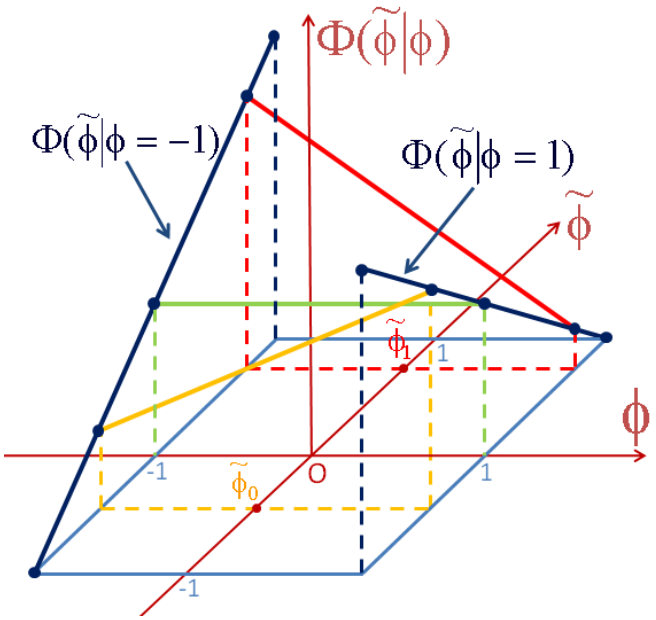


Figure 6.2: The illustration of the relationship between $\Phi(\tilde{\phi}|\phi)$, $\tilde{\phi}$ and ϕ expressed by Equation 6.3. See details in text.

tolerate discrepancies in the model due to sensor noise, changes in the brightness, contrast of the camera, departure from the Lambertian assumption, occlusions and clutter.

6.2.2 Relations to Kolev et al. [94]

In work by Kolev et al. [94], the energy term relative to the measurements and the model is defined as

$$E_{Kolev}(u) = \int (\rho_{bck}(\mathbf{X}) - \rho_{obj}(\mathbf{X}))u(\mathbf{X})d\mathbf{X} \tag{6.4}$$

where $\rho_{bck}(\mathbf{X}) + \rho_{obj}(\mathbf{X}) = 1$ and $\rho_{bck}(\mathbf{X})$ and $\rho_{obj}(\mathbf{X})$ depend on depth maps estimates obtained from different views. For instance, they can be defined as the negative log-likelihood of $\mathbf{X}$ belonging to the object or the background. One of the nice features of this energy formulation is that the two terms $\rho_{obj}(\mathbf{X})$ and $\rho_{bck}(\mathbf{X})$ "compete" to define whether the voxel $\mathbf{X}$ lies inside or outside any of the objects.

By minimizing this energy they obtain a solution u that tends to $+\infty$ on voxels inside objects ($\rho_{obj}(\mathbf{X}) > \rho_{bck}(\mathbf{X})$), and to $-\infty$ outside ($\rho_{obj}(\mathbf{X}) < \rho_{bck}(\mathbf{X})$). By adding a convex energy term that penalizes values of u out of the range $[0, 1]$, u will instead become the indicator function of the inside of the objects.

Now, define the following identities

$$\phi(\mathbf{X}) \doteq 2u(\mathbf{X}) - 1 \tag{6.5}$$

$$\tilde{\phi}(\mathbf{X}) \doteq \rho_{obj}(\mathbf{X}) - \rho_{bck}(\mathbf{X}) = 2\rho_{obj}(\mathbf{X}) - 1 \tag{6.6}$$

in the above energy term. The constraint on $u \in [0, 1]$ becomes $\phi \in [-1, +1]$. Since $\rho_{obj} \in [0, 1]$ by definition, we also have that $\tilde{\phi} \in [-1, +1]$. The resulting energy is identical, up to a constant scale factor, to Equation (6.3). Hence, one can immediately conclude that the term $\rho_{obj} - \rho_{bck}$ also defines an initial estimate of the surface of the objects and the accuracy with which it is obtained determines the overall performance of the reconstruction task.

The main differences of our work and the work of Kolev et al. in [94] are as follows. In [94], for a voxel on the surface, they use the visual hull and its surface normals to predecide which camera can see this voxel. In our work we avoid to use the silhouette informations or visual hull at any stage. The work in [94] does no focus on the robustness to the occlusions or clutter, but that is the main goal of our work.

6.3 Bayesian Photoconsistency

To determine the proxy $\tilde{\phi}$ we obtain depth maps from small groups of nearby views, so that outliers due to occlusions are minimized, and then merge them into a single 3D surface [120, 189]. In our algorithm we follow a similar merging strategy, but try to delay as much as possible hard decisions so as to maximize the amount of information used to take them. In broad terms, the key idea is

to obtain a visibility map from each depth map and then to integrate all the visibility maps together via a robust interpolating function. To illustrate the steps needed, here we will show the computation of a single visibility map. The integration of all the visibility maps will be discussed in the next section.

6.3.1 Photoconsistency Computation

In order to compute a depth map, we need to define how images are generated from the scene. Let $\{I_i : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}^3_+\}_{i=1,\dots,N}$ be a collection of N calibrated color images, $\{\pi_i : \mathbf{V} \subset \mathbb{R}^3 \mapsto \Omega\}_{i=1,\dots,N}$ be perspective projections of a voxel to pixel coordinates in the i-th view, and $\{\mathbf{C}_i \in \mathbb{R}\}_{i=1,\dots,N}$ be the camera centers. Under the Lambertian assumption the intensity measured on the i-th camera sensor can be written as

$$\begin{aligned} I_i(\pi_i[\mathbf{X}]) &= r((\mathbf{X} - \mathbf{C}_i)\lambda_i^* + \mathbf{C}_i) \qquad \text{where} \\ \lambda_i^* &= \arg\min_{\lambda \in [0,\infty)} \{\lambda | \phi(\lambda \mathbf{X} + (1-\lambda)\mathbf{C}_i) = 0\} \end{aligned} \tag{6.7}$$

and $r : V \mapsto \mathbb{R}^3_+$ is the color intensity reflected at a point in space. The above definition formalizes two well-known notions: 1) If two images capture light from the same point in space, the same intensity is observed (photoconsistency); 2) The intensity captured by an image at a pixel $\pi_i[\mathbf{X}]$ depends on the closest surface point to the camera along the ray connecting the camera center $\mathbf{C}_i$ to the point in space $\mathbf{X}$.

Given the i-th view I_i we are interested in computing an estimate of the visibility of a point from this camera. In this case we have that if a point $\mathbf{X}$ on the surface is visible from both the i-th and the j-th camera then

$$I_j(\pi_j[\mathbf{X}]) = I_i(\pi_i[\mathbf{X}])) + \omega \tag{6.8}$$

where ω is sensor noise, which we model with a Laplace distribution. Then we

can write the photoconsistency value between these two views as

$$\rho_{i,j}(\mathbf{X}) = \frac{\sigma}{2I_i(\pi_i[\mathbf{X}])} e^{-\sigma\left|\frac{I_j(\pi_j[\mathbf{X}])}{I_i(\pi_i[\mathbf{X}])}-1\right|} \tag{6.9}$$

where $\frac{1}{\sigma} I_i(\pi_i[\mathbf{X}])$ is the scale parameter. It is reasonable to assume that sensor noise in each view is independent from the other views. Thus the photoconsistency of M views can be computed as the product of individual pairwise photoconsistency terms. The quantity $\rho_{i,j}(\mathbf{X})$ is the probability of photoconsistency and is maximal at the surface when all points are visible and distortions are well modeled by Laplacian noise. Notice that the long tails of the Laplacian distribution allow to compensate for occlusions and other distortions. Furthermore, the degree of tolerance to outliers can be varied by changing the scale parameter. By combining the different views, we obtain

$$\rho_i(\mathbf{X}) = \prod_{j=j_1}^{j_M} \rho_{i,j}(\mathbf{X}). \tag{6.10}$$

where $j_1, \cdots, j_M$ are the indices of the neighbor views of the i-th view. The i-th view is called the central view in this local group and the neighbor views of the i-th view involved in the calculation of Equation (6.10) are called target views in this local group.

Remark The above model rejects outliers similarly to other robust functions that have been suggested in the literature (see, for instance [7, 178]). In practice, the overall behavior is that the photoconsistency term should be as sensitive as possible to small intensity deviations between the views, which are more likely to have been generated by a genuine point on the surface, rather than large intensity deviations, which might have been generated by extremely different phenomena (e.g., occlusions, clutter, and quantization).

The computation of the photoconsistency term Equation (6.10) can be done

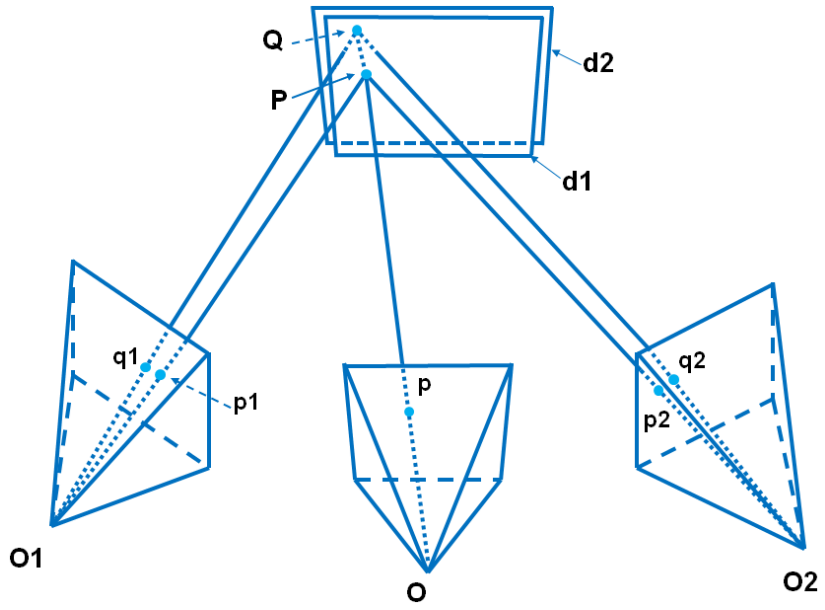


Figure 6.3: The illustration for calculating the photoconsistency probabilities with a plane-sweeping scheme. Details can be seen in the text.

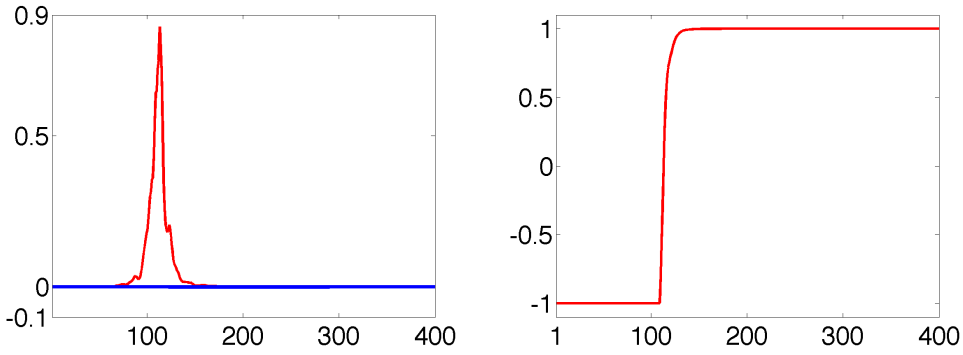


Figure 6.4: The photoconsistency probabilities along two projection rays(left) and the visibility along a projection ray(right).

in a reasonably efficient manner by parsing each point in the volume $\mathbf{V}$. We simply compute the photoconsistency probability independently at each point $\mathbf{X}$ in space. For the sake of simplicity we do not integrate the visibility within windows or slanted planes or compute normalized cross-correlations, although such options are all possible in our framework. Figure 6.3 depicts the procedure for calculating the photoconsistency probabilities with a plane-sweeping scheme.

For a plane parallel to the image plane of the central view(camera O in Figure 6.3) and at certain depth, all the pixels in the central view are backprojected onto this plane and then the corresponding points are projected onto the neighbor views of the central view. The correspondences of the pixel values between the central view and the neighbor views are substituted into Equation (6.10) to calculate the photoconsistency values for each point. This procedure is repeated for each plane located within a certain distance range with respect to the central view. And then for each pixel in the central view, a series of photoconsistency values are computed at different distances along the projection rays. The photoconsistency values along each projection ray are used to decide the depth maps for the corresponding pixel and the visibility functions with respect to each central view are built based on the depth maps.

The photoconsistency values along two projection rays are depicted with a red and a blue curves in the left image in Figure 6.4. It can be seen that, along the red curve, there is a sharp peak with a distinctively high value. On the other hand, the photoconsistency values along the blue curve are all very small such that the blue curve looks like a straight line with zero values everywhere. If there is a sharp peak with a distinctively high value along a ray, we believe that the probability that the corresponding point is located on the surface is high. But if all the photoconsistency values along a projection ray are very small, we will not pick any point along this ray as a surface point. This situation usually happens with four kinds of pixels: the pixels with a large noise, the pixels of background, the pixels with a serious self-occlusion within the local camera group or the pixels

occluded by other objects. For this kind of rays, we set all the voxels along the rays outside of the objects with respect to the current central view. This is expressed in Equation (6.12).

6.3.2 Visibility Mapping

Once Equation (6.10) has been evaluated, we map the photoconsistency probability ρ_i to the visibility of a point $\mathbf{X}$ with respect to the i-th view. Notice that the visibility $\tilde{\phi}_i$ must be a nondecreasing function as we evaluate voxels along a ray from the camera center C_i. We enforce such constraint by considering the integral of $\rho_i(\mathbf{X})$ along the projection ray passing through $\mathbf{C}_i$. We then shift and truncate such function so that the visibility $\tilde{\phi}_i$ is 0 at the depth map, and between -1 and 1 everywhere else. First, for each ray, we compute the location of the depth map

$$\lambda^* \doteq \arg\max_{\lambda\in[0,1]} \rho_i(\mathbf{C}_i + \lambda(\mathbf{X}_{max} - \mathbf{C}_i)) \tag{6.11}$$

where $\mathbf{X}_{max}$ is the furthest point from the camera $\mathbf{C}_i$ along the chosen ray in the volume $\mathbf{V}$. Then, we define the visibility from the i-th view along the ray via the cumulative distribution function of ρ_i, i.e.,

$$\tilde{\phi}_i(\mathbf{C}_i + \mu(\mathbf{X}_{max} - \mathbf{C}_i)) = \begin{cases} \max\left\{-1, \min\left\{1, \int_{\lambda^*}^{\mu} \rho_i(\mathbf{C}_i + \lambda(\mathbf{X}_{max} - \mathbf{C}_i))d\lambda\right\}\right\}, & \rho_{max} \geq \tau \\ -1, \quad \rho_{max} < \tau \end{cases} \tag{6.12}$$

where $\mu \in [0, 1]$ and ρ_{max} is the maximum photoconsistency value along a ray, i.e., the photoconsistency value when $\lambda = \lambda^*$. τ is a predefined threshold parameter. The value of τ is related to several factors. First it is related directly to the parameter σ in Equation (6.9). The larger is the value of σ, the larger is the value of τ. The value of τ is also related to the number of neighbor views of a central view. The more neighbor views of a central view, the smaller value of τ

should be set. The noise level in the input images affects the value of τ too. The higher is the noise level the smaller values should be set to τ. The general rule is that τ should be set a value such that for most pixels projected from background in a central view the corresponding values of ρ_{max} are less than τ and for most pixels projected from foreground in a central view the corresponding values of ρ_{max} are greater than τ.

It is obvious that zero level set of the visibility function of a certain central view is the approximated surface seen from this view. One central view and one view of the zero level set surface of the corresponding visibility function are shown in Figure 6.5. It can be seen that for the pixels occluded by the view-varying occlusions, holes are generated at the corresponding voxels. Because we do not use the silhouettes or visual hull, there are some floating fragments outside of the object. To reconstruct a full scene and at the same time to fill the holes and to eliminate the floating fragments outside the objects, once we have obtained estimates of the visibility $\tilde{\phi}_i$ from each view, we proposed a *Robust interpolating* function to integrate them into a single visibility function $\tilde{\phi}$. This will be presented in the next section.

6.4 Voting Multiple Views in the Presence of Clutter and Occlusions

After the computation of the visibility functions for each central view, we need to integrate them into a single visibility function with a proper voting function. In this chapter we call the final single visibility function the *global visibility* function.

The most common method to integrate depth maps, or visibilities, is to determine which cameras share overlapping views. This can be achieved by using an initial estimate of the surface from the visual hull (i.e., via silhouettes) or from the depth maps themselves. The normals to the surface are then extracted and used

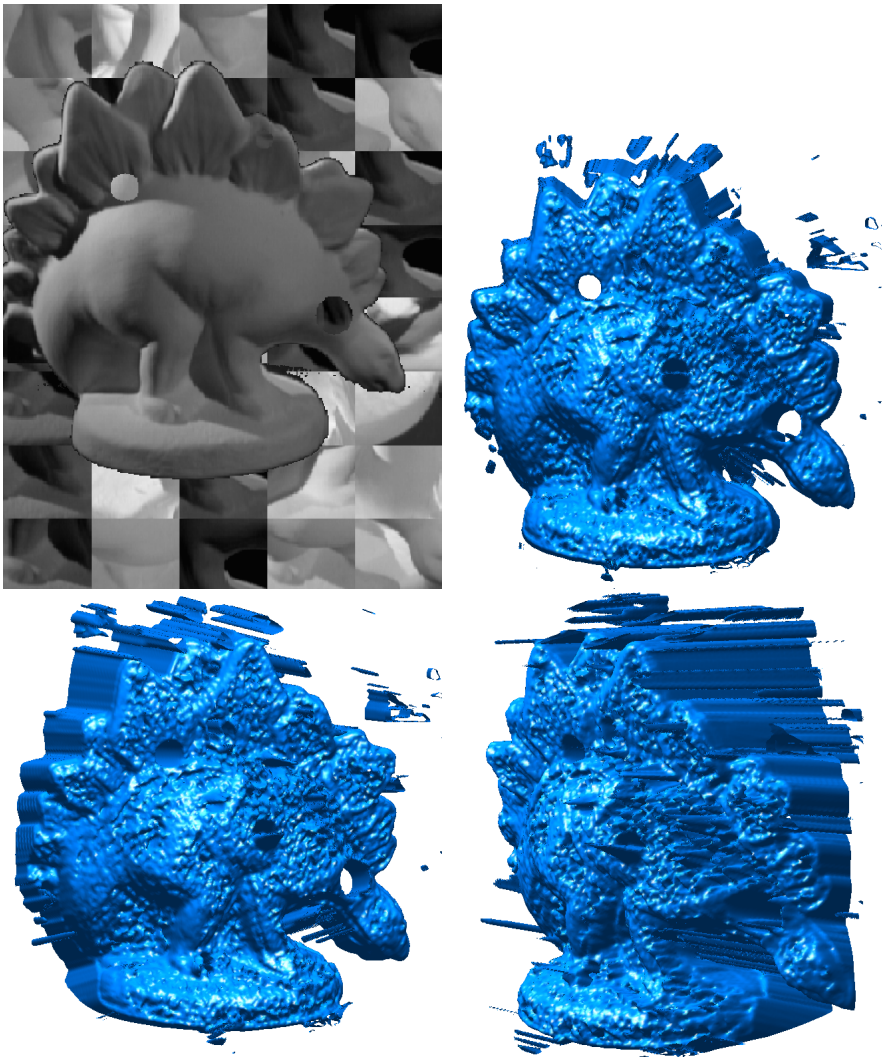


Figure 6.5: The zero level set surface of the visibility function from one central view. Top row: the image of the central view (left) and one view of the zero level set surface of the corresponding visibility function (right). Bottom row: two views of the zero level set surface of the corresponding visibility function.

to determine which cameras are potentially imaging a given voxel. This assumption results in a simple MVS formulation as one only needs to average preselected sets of overlapping depth maps. This, however, comes at a cost, as the normals from an approximate surface estimate could be incorrect and a hard decision on which cameras overlap could lead to averaging depth maps incorrectly.

In this work, we use a different strategy. The procedure of combining the visibility functions can be written as a voting function $f : \mathbb{R}^N \mapsto [-1, 1]$, so that the visibility estimate is given by a certain combination of all the visibilities. It can be expressed as

$$\tilde{\phi}(\mathbf{X}) = f(\tilde{\phi}_1(\mathbf{X}), ..., \tilde{\phi}_N(\mathbf{X})). \tag{6.13}$$

where $\tilde{\phi}_i, 1 \leq i \leq N$, are the visibilities with respect to each central view, which can be computed with Equation (6.12). $\tilde{\phi}$ is the global visibility function. In this general formulation however, it is easier to create a consistent integration of the visibilities and to take into account clutter, occlusions and noise. The function f can be seen as a voting heuristic, where the vote cast by each visibility results in a decision for each voxel in space. Furthermore, in defining f it is important to make sure that some desirable properties are satisfied: for instance, f should be invariant to permutations of the input parameters. Notice that f could be learned from training data so that one could determine the most robust integration method for a given camera configuration. In this work, however, we do not investigate this direction.

6.4.1 Voting Schemes

In this section, we first analyze three choices of the voting schemes and report their performance.

Min Voting Function

This is one of the simplest and most computationally and memory efficient func-

tions as its evaluation can be done recursively as visibilities become available:

$$f(\xi_1, \cdots, \xi_N) = \min_i \xi_i. \tag{6.14}$$

where $1 \leq i \leq N$ is the index of the central views involved in this voting scheme. In this scheme the final visibility value of a particular voxel is chosen as the minimum value among the visibility values of this voxel given by each central view. This scheme comes from the following observations. Given the perfect visibilities, when the visibility of a voxel is 1 with respect to a central view, it just means that this voxel is behind of the objects' surface portion which can be seen by this central view and does not mean it must be inside of the objects. Because it could be beyond the objects' surface at opposite side. But if a voxel has a visibility vale as -1 with respect to one of the visibility functions, it means at least one camera can see it so this voxel must be outside of the objects. For a particular voxel, if the visibility value of it given by one central view is 1 and at the same time its visibility value given by another central view is -1, -1 should be chosen among these two values. In our scheme of visibility mapping, for most voxels, the visibility values are either 1 or -1. But there is transition part around the objects' surface, the voxels within this transition part can take values between -1 and 1 as their visibility values. Given two visibility values of a voxel, we prefer to believe the smaller one. This conclusion is based on the perfect visibilities. So this scheme is very sensitive to noise and unfortunately it is also one of the worst performing ones (see Table 6.1). Notice that this mapping is non linear in the arguments.

Local Mean Voting Function

This interpolating function mimics the choice made by methods that use depth

information to decide which depth maps to average.

$$f(\xi_1, \cdots, \xi_N) = \begin{cases} \frac{1}{N'} \sum_{i=1}^{N'} \xi_i, & \text{if } \exists i_1, i_2, \cdots, i_{N'} : |\xi_{i_k}| \leq \tau \\ \min_{i_k} \xi_{i_k}, & \text{if } \forall i : \ |\xi_i| > \tau \ \& \ \exists i_1, \cdots, i_m : \xi_{i_k} < -\tau, m \geq M \\ \max_i \xi_i, & \text{if } \forall i : \ |\xi_i| > \tau \ \& \ \exists i_1, \cdots, i_m : \xi_{i_k} < -\tau, m < M \end{cases} \tag{6.15}$$

for a small positive constant τ and a predefined integer M.

The ideas of this voting function are as follows. For a voxel, if there are a few views believing that it is close to the surface then the final visibility value of this voxel is set as the average of the visibility values of these views. If all the views believe that a voxel is far away from the surface and for a predefined integer M, there are at least M views believe that it is outside of any object, then the final visibility value of this voxel is set as the minimum among the visibility values of the views which set it outside of any object. If all the views believe that a voxel is far away from the surface and the number of views which can see this voxel is less than M, then its final visibility values is set as the maximum of all the visibility values at this voxel.

Geometric Mean Voting Function

This interpolating function integrates the visibilities from each view by computing a geometric mean.

$$f(\xi_1, \cdots, \xi_N) = \left(\prod_{i=1}^{N} (1 + \xi_i) \right)^{1/N} - 1. \tag{6.16}$$

The idea of this voting function is to reduce the effect of self-occlusion on the visibility. In fact, if a voxel is outside of the objects but it is occluded by a part of the objects to a certain view then this view believes it behind the surface so the visibility value at this voxel with respect to this view is set to 1. This is a

wrong visibility setting because of the self-occlusion. But if there is a view which can see this voxel then the visibility value at this voxel with respect to this view is -1. This is the right visibility setting. Then the final visibility value computed by Geometric mean voting function between these two views is -1. We obtain the right visibility setting.

The comparison and evaluation of the results from different voting functions are shown in Section 6.4.3.

6.4.2 Voting via Interpolation

The last section gave three choices of the voting functions. Besides these, multi-dimensional interpolating functions can be used to combine the visibility functions.

Suppose there are N central views involved in a visibility voting scheme, then for a particular voxel there are N visibility values given by each central view. That is, each voxel is mapped to a N-dimensional vector with the visibility values given by each central view as its elements. For example, if a voxel is mapped to the vector $[1\ 1\ \cdots\ 1]^T$ with all the elements as 1, it means all the central views can not see this voxel. If a voxel is mapped to the vector $[-1\ \ -1\ 0\ 1\ \cdots\ 1]^T$, it means that the first two central views can see this voxel and the third central view believes this voxel is on the objects' surface and all the other central views can not see this voxel. A visibility voting scheme involves to define a rule to map the N-dimensional vectors to scaler values between -1 and 1. Because of the existence of the transition part of the visibility values along an optical ray as mentioned in previous section, the visibility values of some voxels can be any values between -1 to 1. So for a fixed value N, there are still infinite N-dimensional vectors with the visibility values given by each central view as their elements. It is not easy or even possible to define a visibility voting scheme to map each of this kind of vectors to a scaler. So we decide to do the visibility voting in two steps. First

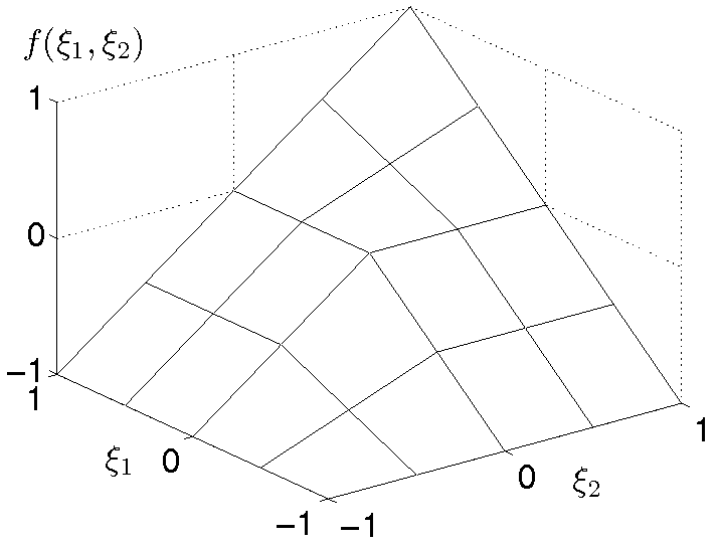


Figure 6.6: The `ideal` voting functions with 2 views ξ_1 and ξ_2.

we define rules to map the vectors, whose elements include only 0, 1 and -1, to scalers and then for the vectors with other combinations, the corresponding values are given by the N-dimensional interpolating function. More explicitly, we define the known points of f at the finite set of locations in the N-dimensional grid $\{-1, 0, 1\} \times \cdots \times \{-1, 0, 1\}$ and for the points away from the grid its visibility value is given by the N-dimensional interpolating function (an example of 2-dimensional grid $\{-1, 0, 1\} \times \{-1, 0, 1\}$ is shown in Figure 6.6). In this section, two interpolating functions are discussed. One is the *Ideal interpolating* function and the other one is the *Robust interpolating* function. The difference between them is how to define the corresponding values for the grid points. That is how to define the known points for the interpolations.

Ideal Interpolating Function

The known points of the Ideal interpolating function on the N-dimensional grid $\{-1, 0, 1\} \times \cdots \times \{-1, 0, 1\}$ are defined as follows:

$$f(\xi_1, \cdots, \xi_N) = \begin{cases} -1, & \text{if } \exists\ i:\ \xi_i = -1 \\ 1, & \text{if } \xi_i = 1 \quad \forall\ i \\ 0, & \text{at all other locations} \end{cases} \tag{6.17}$$

The idea of this interpolating function is as follows. For a grid point on the N-dimensional grid, if any of the input visibility values is -1 then also f maps to -1. This corresponds to the situation that a voxel is set outside of the objects if there is one central view which can see it. If all input visibility values are 1 then f is also set to 1. This corresponds to the situation that a voxel is set inside of the objects if noun of the cameras can see it. All the other grid points are defined on the surface. Obviously, this interpolating function works well in ideal conditions, that is, all the input visibilities are computed perfectly.

Robust Interpolating Function

The above mentioned voting schemes, including the Ideal interpolating function, work well when the input visibilities are perfect or with small errors. In fact we can never obtain perfect visibilities, especially, when in the presence of clutter and occlusions. See the second image in Figure 6.5, it is far from perfect. With this kind of input data, as it can be seen in the next section, none of them works fine. A robust voting scheme which tolerates conflicting terms are essential. For this purpose we introduce the *Robust interpolating* function.

As in the case of the *Ideal interpolating* function, we first define the known points of f at the finite set of locations in the N-dimensional grid $\{-1, 0, 1\} \times \cdots \times \{-1, 0, 1\}$ as follows:

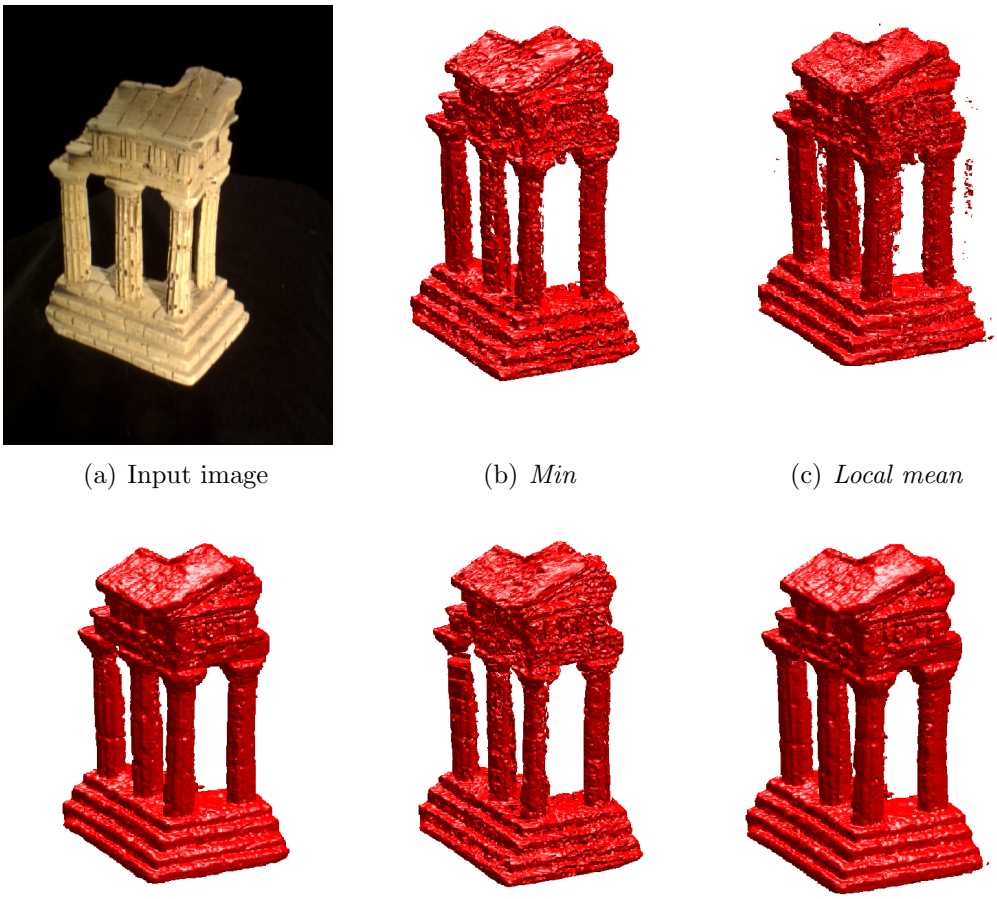
$$f(\xi_1, \cdots, \xi_N) = \begin{cases} -1, & \text{if } \exists\ i_1, \cdots, i_M : \xi_{i_k} = -1 \\ 1, & \text{if } \exists\ i_1 \cdots i_{N-M+1} : \xi_{i_k} = 1 \\ 0, & \text{at all other locations} \end{cases} \tag{6.18}$$

The meaning of the above definition is that we set a voxel outside of the objects if at least M views agree that this voxel is outside of the objects. If we want to set a voxel inside the object we need at least $N - M + 1$ visibilities agreeing that it is inside the object. For all the other cases, we set the voxel on the surface of the object.

The value of M depends on the number of the input visibilities. Based on our experiments, the integer value of M can be chosen up to 20% of the number of input visibilities. If the value of M is chosen about 20% of the number of input visibilities, it means it can tolerate about 20% of the wrong decisions. This is particularly effective in the presence of view-varying clutter and occlusions, where several visibilities may be incorrect at some locations.

method names	Robust interpolating	Geometric mean	Local mean	Ideal interpolating	Min
accuracy0%	9.01e-5	4.08e-4	1.58e-4	1.65e-4	3.03e-4
coverage0%	100%	99.42%	99.88%	99.62%	98.83%
accuracy1%	1.03e-4	1.03e-4	1.81e-4	3.28e-4	7.72e-4
coverage1%	100%	98.29%	99.70%	98.37%	93.27%
accuracy2%	1.58e-4	6.51e-4	5.06e-4	8.93e-4	1.7e-3
coverage2%	99.99%	96.53%	95.75%	90.27%	75.59%
accuracy3%	2.75e-4	7.79e-4	1.0e-3	1.5e-3	2.5e-3
coverage3%	99.73%	94.87%	89.83%	78.54%	56.20%
accuracy5%	6.43e-4	9.49e-4	2.4e-3	3.1e-3	4.6e-3
coverage5%	95.66%	92.72%	76.73%	49.05%	24.48%
accuracy occlusion	1.4e-3	3.3e-3	4.8e-3	4.7e-3	4.6e-3
coverage occlusion	89.04%	67.50%	48.81%	46.92%	46.99%

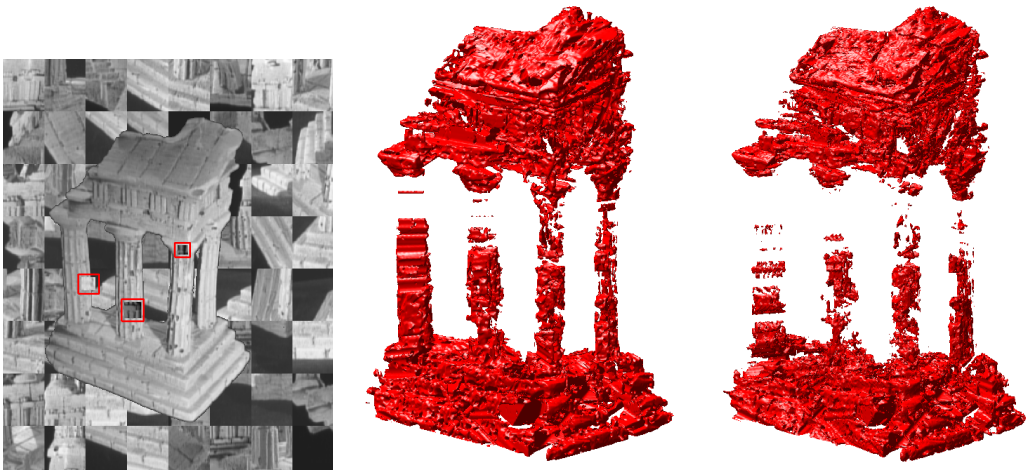
Table 6.1: Performance on the *Sphere* synthetic dataset with 5 different voting functions: *Robust interpolating*, *Geometric mean*, *Local mean*, *Ideal interpolating* and *Min*. The percentages in the first column of the table indicate the noise levels. The unit of the accuracy is meter. The last two rows show the performance when view-varying occlusions (a randomly placed disk) are present. See details in the text for the meanings of accuracy and coverage.



(a) Input image (b) *Min* (c) *Local mean*

(d) *Geometric mean* (e) *Ideal interpolating* (f) *Robust interpolating*

Figure 6.7: Comparison of the reconstructed visibilities $\tilde{\phi}$ (without smoothing) of five voting function choices with the *temple* dataset. First row: the first image is one of the input images. The last two are one view of the zero level set of the integrated visibility functions from *Min* and *Local mean* voting functions respectively. Second row(left to right): one view of the zero level set of the integrated visibility functions from *Geometric mean* voting function, *Ideal interpolating* and *Robust interpolating* functions respectively.



(a) Input image (b) *Min* (c) *Local mean*

(d) *Geometric mean* (e) *Ideal interpolating* (f) *Robust interpolating*

Figure 6.8: Comparison of the reconstructed visibilities $\tilde{\phi}$ (without smoothing) of five interpolating function choices with the *temple* dataset in the presence of clutter and occlusions. The image arrangement is the same as in Figure 6.7. The only difference in these experiments is the input images with clutter and occlusions (the squares highlighted in the red windows).

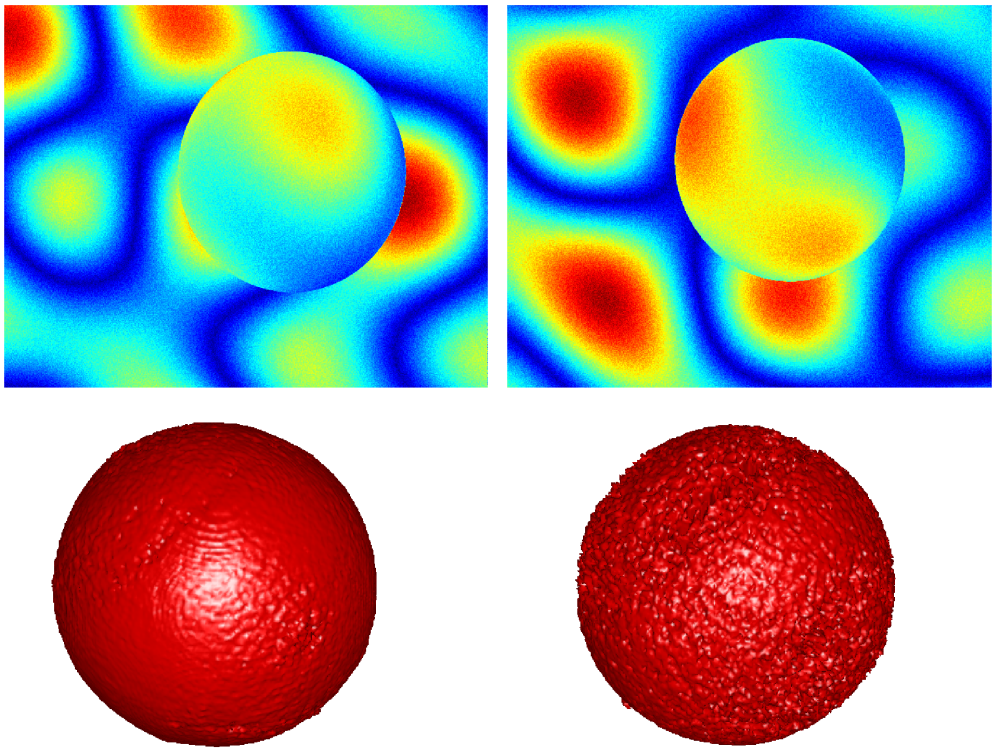


Figure 6.9: Comparison of the reconstructed results of the *Sphere* synthetic dataset between *Robust interpolating* function and *Min* voting function without view-varying occlusions at 3% noise level. Top row: Two of the 60 input images. Bottom row: The reconstructed 3D model with *Robust* interpolating function (left) and *Min* voting function (right).

6.4.3 Evaluations of the Different Voting Functions

In this section, the comparisons and valuations of the performance of different voting functions are shown.

First we show a direct comparison of the 5 choices of the voting functions on the Middlebury *Temple* dataset in Figure 6.7. As one can see, the results from the *Min*, *Local mean* voting functions and *Ideal interpolating* function are quite poor compared with the result from the *Robust interpolating* function. The only one that is comparable with the *Robust interpolating* function is the *Geometric mean* voting function. But if we observe carefully, we will find that the upper part of two pillars are killed too much in the result from the *Geometric mean*

voting function. So the *Robust interpolating* function is more successful than the other functions in preserving more of the surface and at the same time in avoiding artifacts due to incorrect visibility estimates. The advantages of the *Robust interpolating* function are more obvious when the input data with view-varying clutter and occlusions. This can be seen in Figure 6.8 where the input data are the modified *Temple* dataset in the presence of view-varying clutter and occlusions as shown with the first image. The other images in Figure 6.8 are the zero level sets of the integrated visibility functions from the corresponding voting functions. It can be seen that while the result from the *Geometric mean* voting function is still better than the results from the *Ideal interpolating*, *Local mean* and *Min* voting functions, it is much poorer than that from the *Robust interpolating* function.

We also run several experiments on synthetic *Sphere* dataset where, in different experiments, we consider images with different levels of noise and with or without view-varying occlusions.

In Figure 6.9 we show some results of the experiments from *Robust interpolating* function and *Min* voting function on synthetic *Sphere* dataset. In these experiments, the input images are with 3% noise and without view-varying occlusions. These experiments demonstrate the effectiveness of the *Robust interpolating* function in dealing with the noise in the input images.

The evaluations of the experimental results on synthetic *Sphere* dataset are shown in Table 6.1. In these evaluations, the *accuracy* is computed as the average distance of the reconstructed surface points to the ground truth. For a point on the true surface, if there is a point on the reconstructed surface with distance less than 0.2 millimeters to it, we would say this point is covered by the reconstructed surface. Otherwise, this point will not be counted in the coverage.

It can be seen from Table 6.1 that, with the data without view-varying occlusions or noise, although the *Robust interpolating* function does slightly better than all the others, any voting function can do a good job. As the noise level

up to 5%, the results from the *Ideal interpolating*, *Local mean* and *Min* voting functions are quite poor. This is the same as the results from the *Temple* data set. It means they are not robust to noise. At this noise level the result from the *Geometric mean* voting function are acceptable, but the best job are from the *Robust interpolating* function. In the presence of view-varying occlusions, the coverage of the reconstructed surface from the *Robust interpolatin* function is close to 90%. For others, they are all below 70%. So in this work, we propose to use the *Robust interpolating* function to do the visibility voting.

One shortcoming of the proposed approach is that the *Robust interpolating* function grows in complexity with the number of central views that it integrates. While this is perfectly tolerable for medium-size reconstructions, it is unmanageable for large-size reconstructions. This challenge can be solved by parallel computing, but in this work we do not investigate this direction.

6.5 Numerical Implementation

After defining all the functions and parameters for the minimization problem defined with Equation (6.1), we can solve it by first computing the Euler-Lagrange equations and then using a numerical scheme to solve them. In the case of the discrepancy function given with Equation (6.2) we have

$$\begin{aligned} \nabla E[\phi(\mathbf{X})] = \quad & \frac{\phi(\mathbf{X}) - \tilde{\phi}(\mathbf{X})}{|\phi(\mathbf{X}) - \tilde{\phi}(\mathbf{X})|} - \alpha \nabla \cdot \left(\Psi(\tilde{\phi}(\mathbf{X})) \frac{\nabla \phi(\mathbf{X})}{|\nabla \phi(\mathbf{X})|} \right) \\ & + \beta \theta'(\phi(\mathbf{X})) = 0 \qquad \forall \mathbf{X} \in V \end{aligned} \tag{6.19}$$

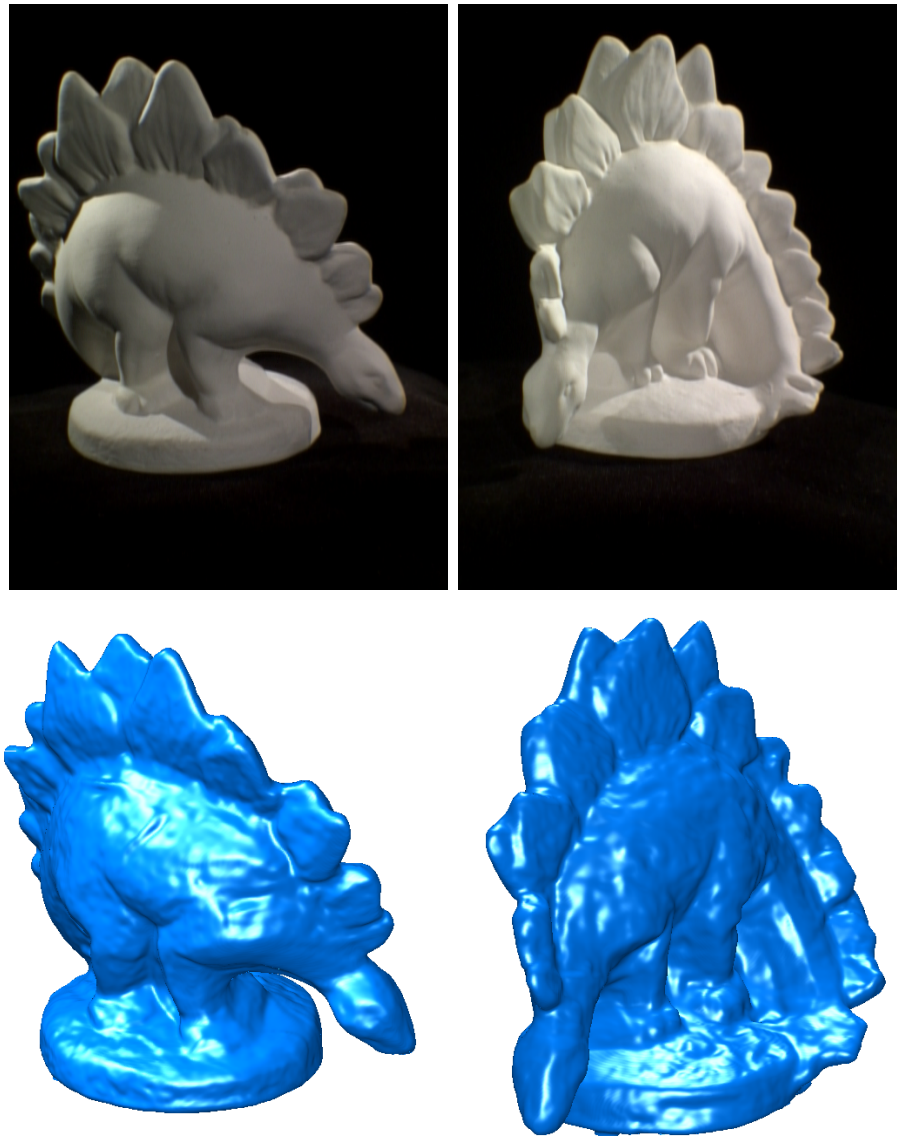


Figure 6.10: The reconstruction of *Dinosaur* dataset. The first row shows two input images. The second row shows two views of the reconstructed 3D model obtained with the *Robust interpolating* function.

and in the case of the discrepancy function given with Equation (6.3) we have

$$\begin{aligned} \nabla E[\phi(\mathbf{X})] \quad &= -2\tilde{\phi}(\mathbf{X}) - \alpha\nabla \cdot \left(\Psi(\tilde{\phi}(\mathbf{X})) \frac{\nabla\phi(\mathbf{X})}{|\nabla\phi(\mathbf{X})|} \right) \\ &+\beta\theta'(\phi(\mathbf{X})) = 0 \qquad \forall \mathbf{X} \in V. \end{aligned} \tag{6.20}$$

Figure 6.11: The reconstruction of *Temple* dataset. The first row shows two input images. The second row shows two views of the reconstructed 3D model obtained with the *Robust interpolating* function.

These equations could be solved via linearization and successive over-relaxation (or other iterative solvers for linear systems). However, we find that a gradient descent scheme works quite efficiently on these functionals and most of the processing time is actually spent in the pre-computation of the estimate $\tilde{\phi}$. Notice that we prevent any division by zero by introducing a small positive constant.

The solution of the Euler-Lagrange equations via a gradient descent is given by

$$\phi(\mathbf{X}, t+1) = \phi(\mathbf{X}, t) - \varepsilon \nabla E[\phi(\mathbf{X}, t)] \tag{6.21}$$

for a small step $\varepsilon > 0$. Starting from any initial condition, this iterative scheme will converge to the global minimum of functional (6.1). Notice that the smoothness term is formulated as total variation and therefore it tends to yield 3D surfaces that are piecewise smooth.

Data set	completeness	accuracy	#views
dinosaur(robust)	99.3%	0.60 mm	60
temple(robust)	98.6%	0.76 mm	56
dinosaur(geometric)	95.7%	1.94 mm	60
temple(geometric)	89.6%	1.45 mm	56

Table 6.2: Performance on the *Dinosaur* and *Temple* datasets with two different voting functions *Robust interpolating* and *Geometric mean*. Notice that the results from the *Robust interpolating* function are at the same level as the state-of-the-art in MVS.

6.6 Experiments on the First Method

To demonstrate the effectiveness of the proposed method we use the two multiview stereo data sets publicly available at the Middlebury website [1]: *Dinosaur* and *Temple* datasets. The performance results that we have obtained with the original datasets from the *Robust interpolating* and *Geometric mean* voting functions are shown in Table 6.2. From the table, first we can see that although the zero level set of the visibility function from the *Geometric mean* voting function displayed in Figure 6.7 are comparable to that from the *Robust interpolating* function, but from the evaluation, the *Robust interpolating* function works much more accurately than the *Geometric mean* voting function. Second, our results are not the best in the *Middlebury evaluation list* (see Figure 6.12), but they are

Sort By	Temple Full 312 views Acc [mm]	Temple Full 312 views Comp [%]	Temple Ring 47 views Acc [mm]	Temple Ring 47 views Comp [%]	Temple Sparse 16 views Acc [mm]	Temple Sparse 16 views Comp [%]	Dino Full 363 views Acc [mm]	Dino Full 363 views Comp [%]	Dino Ring 48 views Acc [mm]	Dino Ring 48 views Comp [%]	Dino Sparse 16 views Acc [mm]	Dino Sparse 16 views Comp [%]
Auclair			0.86	96.2	1.03	92.5			0.62	96.7	0.74	96.8
Bradley			0.57	98.1	0.48	93.7			0.39	97.6	0.38	94.7
Campbell	0.41	99.9	0.48	99.4	0.53	98.6						
Chang			0.54	99.0	0.73	94.5			0.51	94.6	0.66	89.9
Continuous Probab			1.89	92.1					2.61	91.4		
Delaunoy					0.73	95.9					0.89	93.9
Deng			0.54	98.5							0.43	97.8
Depth Fusion			0.53	99.5	0.72	96.8			0.46	99.5	0.42	97.8
Furukawa	0.65	98.7	0.58	98.5	0.82	94.3	0.52	99.2	0.42	98.8	0.58	96.9
Furukawa 2	0.54	99.3	0.55	99.1	0.62	99.2	0.32	99.9	0.33	99.6	0.42	99.2
Furukawa 3	0.49	99.6	0.47	99.6	0.63	99.3	0.33	99.8	0.28	99.8	0.37	99.2
Gargallo			0.88	84.3	1.05	81.9			0.6	92.9	0.76	90.7
Goesele	0.42	98.0	0.61	86.2	0.87	56.6	0.56	80.0	0.46	57.8	0.56	26.0
Goesele 2007	0.42	98.2					0.46	96.7				
Guillemaut	0.43	99.0	0.71	97.6	0.86	96.2	0.35	100	0.58	99.5	0.68	98.0
Habbecke	0.66	98.0					0.43	99.7				
Hernandez	0.36	99.7	0.52	99.5	0.75	95.3	0.49	99.6	0.45	97.9	0.6	98.5
Hongxing	0.83	95.7	0.79	96.3	0.97	93.9	0.62	96.3	0.5	99.1	0.52	98.4
Hornung	0.58	98.7					0.79	95.1				
Jancosek-3DIM09	0.65	85.8	0.7	78.9	0.59	74.9	0.91	73.8	0.71	76.6	0.66	74.9

Figure 6.12: Middlebury evaluation list.

at the same level as the state-of-the-art in MVS. One main reason affects the accuracy of the reconstructed model is that when we generate the depth maps we compute the photoconsistency values independently for each point not within a window. This leads to a very noisy depth map as shown before. We compute the photoconsistency values in this way for efficiency and at the same time it shows that the proposed method is robust to noise.

Two input images of the *Dinosaur* dataset and two views of the reconstructed model from the *Robust interpolating* function are shown in Figure 6.10 and two input images of the *Temple* dataset and two views of the reconstructed model from the *Robust interpolating* function are shown in Figure 6.11. The reconstructions obtained with the two discrepancies Equation (6.3) and Equation (6.2) are virtually identical; so, we only display the results obtained for Equation (6.3).

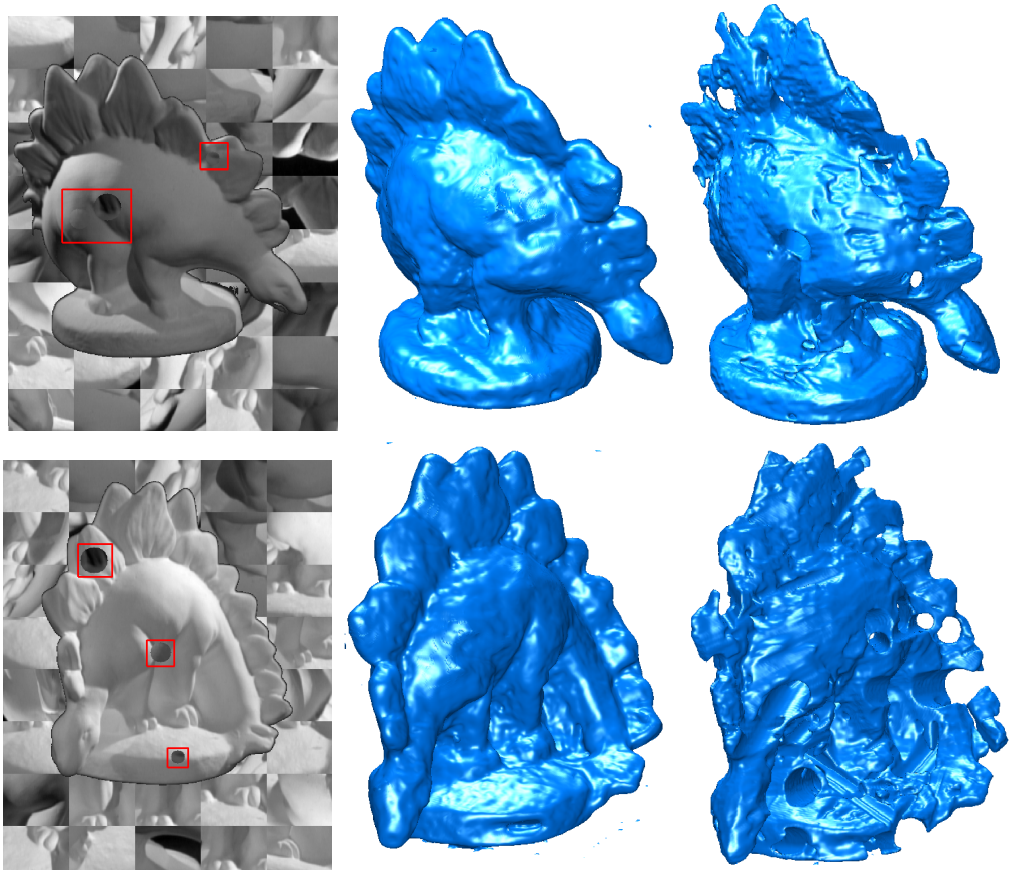


Figure 6.13: Input images and reconstructed results from the *Dinosaur* dataset in the presence of view-varying occlusions and clutter. The images in the first column are two central views with the occlusions included in the red boxes. The images in the second column are two views of the reconstructed model from the *Robust interpolating* function and the images in the last column are two views of the reconstructed model from the *Local mean* voting function.

In Figure 6.13 and 6.14 we show the input images and corresponding reconstructed results of *Dinosaur* and *Temple* datasets with view-varying occlusions and clutter. It can be seen that in the presence of view-varying occlusions, the reconstruction results from the *Robust interpolating* function are still comparable to the ones obtained with uncluttered images, but the results obtained from other interpolating functions are much worse. Here we just display the results from the *Local mean* voting function for comparison.

The input images in the experiments are generated from the Middlebury *Di-*

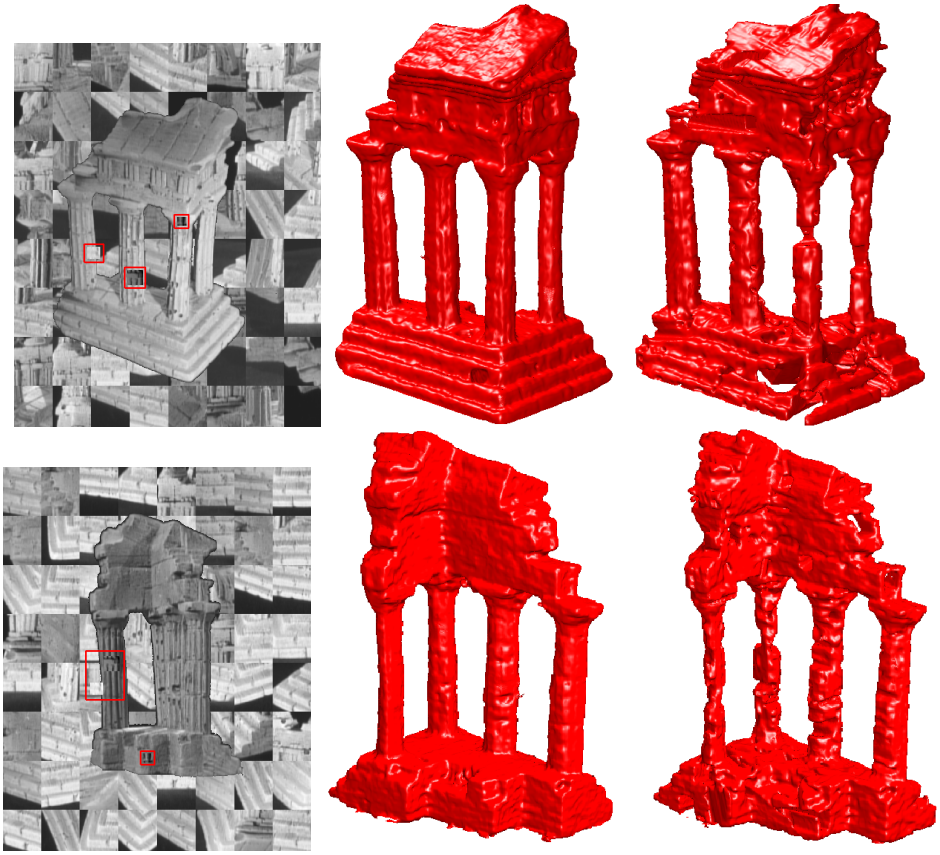


Figure 6.14: Input images and reconstructed results from the *Temple* dataset in the presence of view-varying occlusions and clutter. The images in the first column are two central views with the occlusions included in the red boxes. The images in the second column are two views of the reconstructed model from the *Robust interpolating* function and the images in the last column are two views of the reconstructed model from the *Local mean* voting function.

nosaur and *Temple* datasets as follows: The background images are generated by tiling image patches which are obtained by cutting a patch at a random location in a randomly chosen image in the corresponding image group. With the help of an approximated silhouette generated from the original image, a cluttered background is attached to the original image. And then the three occluders are attached onto each of them at random locations.

We used a Mac Pro 8-core 3.2GHz in our experiments. The reconstructions are defined on a volume of $359 \times 301 \times 307$ voxels for the *Dinosaur* dataset and $465 \times 301 \times 219$ voxels for the *Temple* dataset and are produced in less than 30

minutes after the computation of the visibility $\tilde{\phi}$. The time taken for computing the visibility function depends on how many reference views have been used. If 12 reference views are used in *Dinosaur* dataset, it takes about 245 minutes for *Robust interpolating* and *Ideal interpolating* functions and about 72 minutes for *geometric*, *local* and *min* interpolating functions.

6.7 An Occlusion-Robust Surface Integration Formulation via Narrow Bands in Multiview Stereo

The problem we are going to solve with this method is the same as the one solved by the preceding method. Here we give the energy formulation directly without stating the problem again.

The energy formulation proposed in this method has the following form

$$\begin{aligned}\hat{\phi} &= \arg\min_{\phi} E(\phi, c_1, c_2) \\ &\doteq \mu \int_V \delta(\phi(X))|\nabla\phi(X)|dX \\ &+\lambda_1 \int_V |\Psi(X) - c_1|^2 H(\phi(X))H(\omega(X))dX \\ &+\lambda_2 \int_V |\Psi(X) - c_2|^2 (1 - H(\phi(X)))H(\omega(X))dX \end{aligned} \tag{6.22}$$

Where function ϕ has the same meaning as in Equation (6.1), function H is the Heaviside function and δ is the *Dirac delta* function. $\mu \geq 0$, $\lambda_1, \lambda_2 > 0$ are fixed parameters. N is the number of central views. $\omega(X) = \sum_{i=1}^{N} \omega_i(X) - \nu$, with ν a nonnegative integer and $\omega_i(X)$ represents the weighting volume for the i^{th} central view. In this work we use a simple binary weighting, $\omega_i(X) \in \{0, 1\}$. For the i^{th} central view, within a narrow band around the approximated surface $\omega_i(X) = 1$ and for other points $\omega_i(X) = 0$ (the details of the weight setting are given in the

next section). In the experiments shown in this chapter we set $\nu = 1$. This means that a point needs to be seen by at least two views to be reconstructed.

$\Psi(X)$ is computed with the following expression:

$$\Psi(X) = \begin{cases} \Phi(X)/(\omega(X) + \nu), & \omega(X) > 0 \\ 0, & \omega(X) = 0 \end{cases} \tag{6.23}$$

where $\Phi(X) = \sum_{i=1}^{N} \omega_i(X)\tilde{\phi}_i(X)$ and $\tilde{\phi}_i(X)$ is the visibility function of the i^{th} central view as same as that in the preceding method.

The values of c_1 and c_2 of a point X are related to the visibility values at it and proportional to the number of cameras which have non-zero weighting at it. For different points, the visibility values and the number of cameras of non-zero weightings could be different. So we compute them with the following expressions:

$$c_1(\phi) = \frac{\int_V \Psi(X)H(\phi(X))H(\omega(X))dX}{\int_V H(\phi(X))H(\omega(X))dX} \tag{6.24}$$

$$c_2(\phi) = \frac{\int_V \Psi(X)(1 - H(\phi(X)))H(\omega(X))dX}{\int_V (1 - H(\phi(X)))H(\omega(X)).dX} \tag{6.25}$$

The first term in Equation (6.22) is a regularization term to penalize the surface area. The last two terms are data fidelity terms. In practice the Heaviside function H is replaced by a regularized vision H_ϵ . Here we use the one presented in Chapter 4, Section 4.5, and suggested in [31].

$$H_\epsilon(z) = \frac{1}{2}\left(1 + \frac{2}{\pi} arctan(\frac{z}{\epsilon})\right) \tag{6.26}$$

The corresponding regularized vision of *Dirac delta* function has the following form

$$\delta_\epsilon(z) = \frac{\epsilon}{\pi(\epsilon^2 + z^2)} \tag{6.27}$$

In Figure 6.15, two plots of the regularized vision of Dirac delta function are

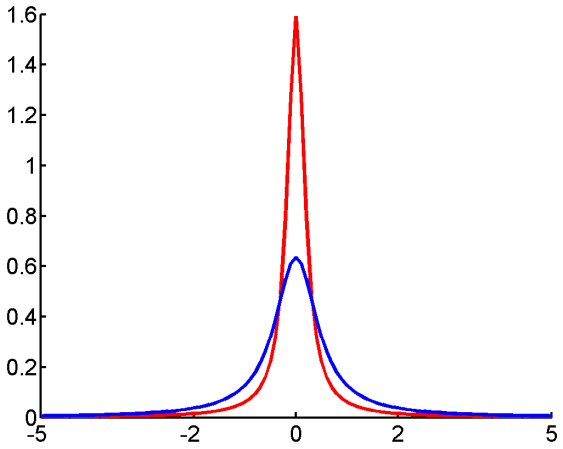


Figure 6.15: Two plots of the regularized vision of Dirac delta function with $\epsilon = 0.2$ (red) and $\epsilon = 0.5$ (blue).

shown. It can be seen that the larger is the value of ϵ and the heavier tails the regularized Dirac delta function has. So if we use a regularized Dirac delta function with a larger ϵ, say 2, in the Equation (6.22), the formulation will take a quite wide band around the zero level set of ϕ into account. This helps the convergence to the global minimum.

Equation. (6.22) can be solved by first computing the Euler-Lagrange equation and then employing a numerical scheme. The Euler-Lagrange equation of the proposed energy functional is:

$$\Delta E[\phi] = \delta_\epsilon(\phi)\left[\mu div\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \lambda_1(\Psi - c_1)^2 + \lambda_2(\Psi - c_2)^2\right] = 0, \forall X \in V \tag{6.28}$$

The iterative step of a gradient descent problem for solving the Euler-Lagrange equation is given by:

$$\phi(X, t+1) = \phi(X, t) - \Delta t \nabla E[\phi(X, t)] \tag{6.29}$$

for a small step Δt.

6.8 Extraction of the Narrow Bands

Based on the visibility functions computed for each central view given in the preceding method presented in this chapter, we explain how to extract a narrow band around the approximated surface from each visibility function. In fact we extract the narrow bands by setting weights to each point along each optical ray starting from the center of any central camera.

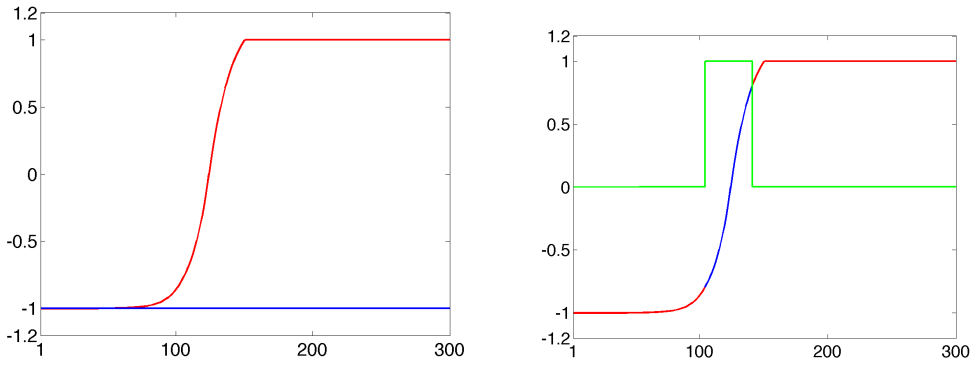


Figure 6.16: The visibility values along two optical rays (left) and the weight setting along a ray. See details in the text.

The visibility value along two rays are shown in the left image of Figure 6.16. Along the blue line, all the visibility values are small (very close to -1), and therefore the corresponding ray does not hit the surface. Along the red curve, the visibility values change from -1 to 1 and the point with visibility 0 is the suggested first intersection of the corresponding ray with the surface. There is a transition around the approximated surface point with values from -1 to 1. The number of voxels in this transition can be changed by tuning the parameter σ in Equation (6.9). To take a narrow band around the approximated surface, the following expression is used to define the weights along the ray:

$$\tilde{\omega}_i(X) = \begin{cases} 1, & \xi < \tilde{v}_i(X) < \beta \\ 0, & otherwise \end{cases} \tag{6.30}$$

where $-1 < \xi < 0$ and $0 < \beta < 1$. These parameters determine the width of the band. As for how to choose values for ξ and β, the general rules are as follows. First, the absolute values of ξ and β should be close to each other. This will help to keep the narrow band approximately symmetric to the reconstructed surface. Second, the absolute values of ξ and β should be neither close to 0 nor to 1. If the absolute values of ξ and β are too small (very close to zero), the narrow band will be too thin leading to holes on the reconstructed surface. On the other hand, if the absolute values of ξ and β are too large (very close to 1), the narrow band can be too wide and this will also lead to some problems. For example, when we reconstruct an object with a extensive thin surface, if the narrow band is too wide, it could be thicker than the surface itself, then the narrow bands generated for reconstructing this side of surface could affect the reconstruction of the surface on the opposite side. So without considering other constraints, $\xi = -0.5$ and $\beta = 0.5$ could be a safe choice.

An example of the weights along a ray is shown with the right image in Figure 6.16. In this image, the green curve shows the weights along this ray and the red curve shows the original visibility values along the same ray. The blue part on the red curve consists of the visibility values of the points involved in the narrowband along this ray.

After setting the weights along each optical ray starting from the center of the i^{th} central camera, the weighted volume $\sigma_i(X)$ is built with respect to this central camera. A slice of the weighted volume with respect to a central view in the *Sphere* dataset is shown in Figure 6.17.

6.9 Surface Estimation

After obtaining the weighting volumes ω_i $(i = 1, \cdots, N)$, the function $\Psi(X)$ defined in the energy formulation (6.22) can be computed with Equation (6.23).

Figure 6.18 shows a schematic of a slice through the function $\Psi(X)$. In this

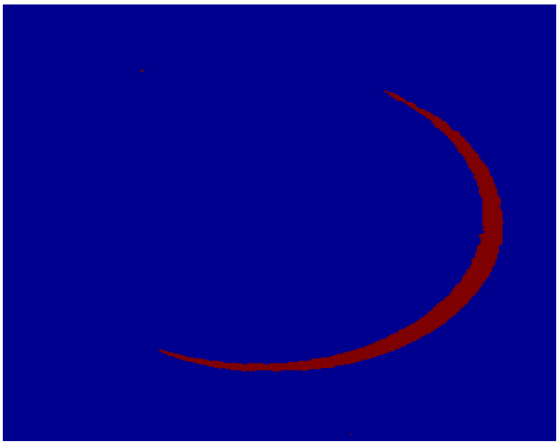


Figure 6.17: A slice of the weighted volume (narrow band) with respect to a central view in *Sphere* dataset.

schematic, the part between the green and red curves is the part within the narrow band. The value at a point A outside of the green curve is 0, the value at a point B inside of the red curve is also 0, the value at a point C between the green and blue curves is positive and the value at a point D between the blue and red curves is negative. So the blue curve is the zero level set, i.e. the reconstructed surface. If the Matlab routine *isosurface* is used to generate the zero level set from the slice shown in Figure 6.18, the zero isosurface includes not only the blue curve, the reconstructed surface, but also the red curve, the interface between negative values and zero values.

Figure 6.19 shows a slice of the reconstructed volume of the *Sphere* dataset. The two green curves shown in this figure are the zero contours generated by the Marching Cubes method. The outer one is on the reconstructed surface, the inner one is on the interface between the narrow band around the approximated surface and the interior part of the sphere. To evaluate the reconstructed results or to render the reconstructed surfaces with other software, it is necessary to extract the surface points from the zero level set. Here we briefly explain the procedure

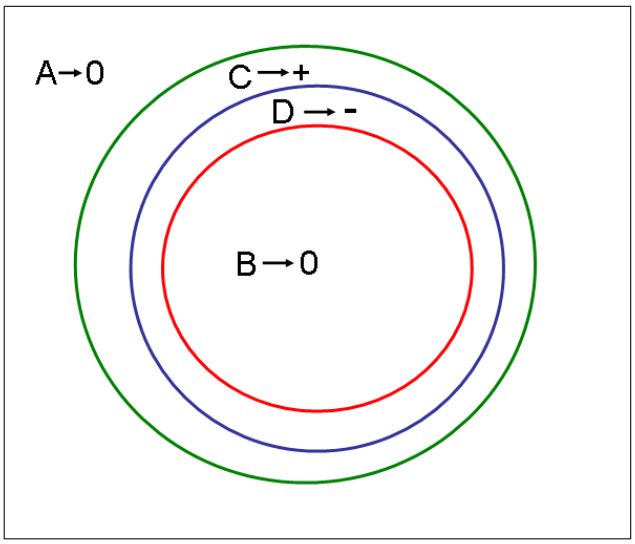


Figure 6.18: A schematic of a slice of the function $\Psi(X)$ defined in Equation (6.22).

of extracting the surfaces from the zero level set obtained from the algorithm proposed in this work.

First vertices and the facets of the zero level set are generated by the Marching Cubes method. For a certain central view, we sort the facets by the distances from the centers of the facets to the camera center in the near to far manner. Then, the facets will be projected onto the image plane in the sorted sequence. If a facet is projected onto a region which has been covered by the projections of the previous facets, then it is not on the object surface with respect to this central view. We say that this facet has no contribution to this view. This procedure is repeated to each central view. The reconstructed surface comprises the facets which contribute to at least one central view. An example is shown in Figure 6.20. The left image shows the zero level set of the visibility values of a central view. The extracted surface is shown in the right image.

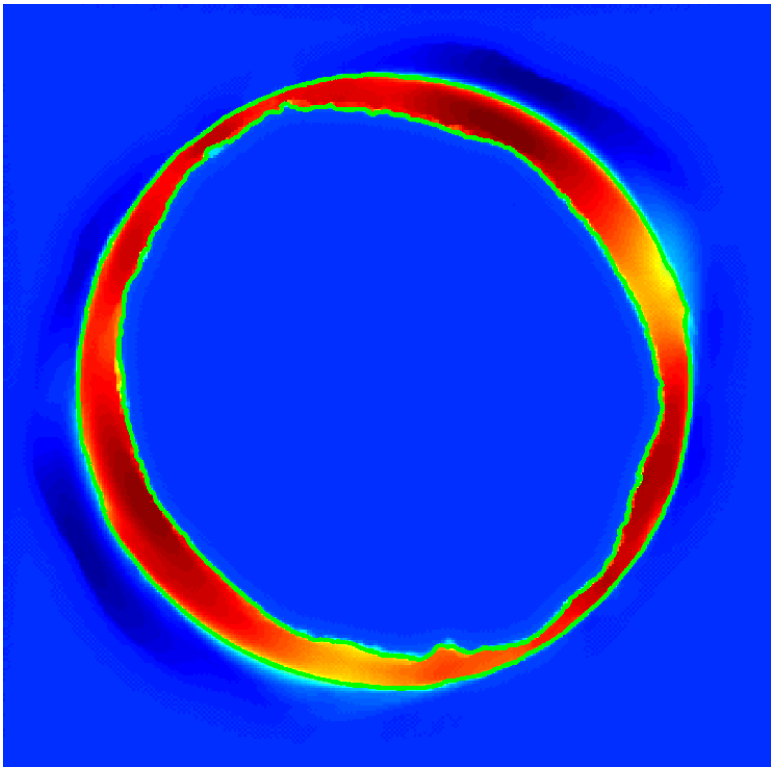


Figure 6.19: A slice of the reconstructed volume in *Sphere* dataset. The two green curves are the zero contours.

6.10 Experiments on the Second Method

We test the algorithm proposed in this paper by synthetic data and real data with synthetic background and occlusions. As we have discussed in Chapter 4, the values of λ_1 and λ_2 should be equal to each other to avoid fevering any of the two data fidelity terms. The ratio between μ and λ_1 (or λ_2) tunes the work of the smoothing term against the work of the data fidelity terms. So we can fix the values of λ_1 and λ_2 and tune the values of μ only in the experiments. In all the experiments, the parameters λ_1 and λ_2 in Equation 6.22 are set to 1. For generating the images displayed in Figure 6.22 and 6.23, parameter μ is set to 0.3.

In the numerical implementation, we tried different numerical schemes such as ENO2 and WENO. Because there is no sharp corners on the reconstructed

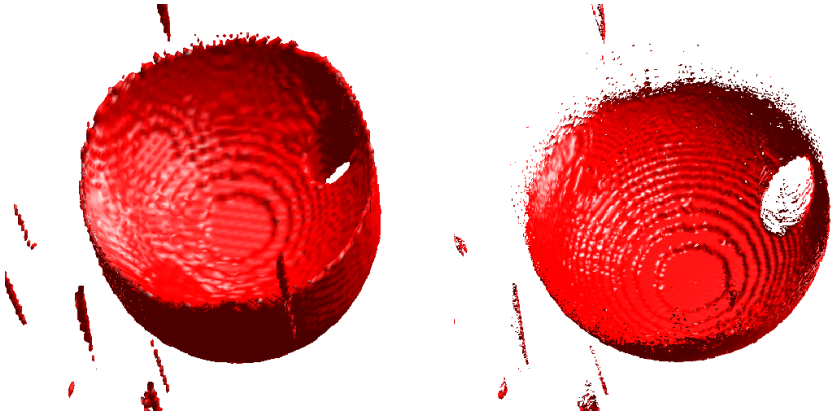


Figure 6.20: The zero level set of the visibility values of one central view and the corresponding extracted surface.

surface, we can no tell much difference of the reconstructed surfaces visually. The displayed results in Figure 6.22 and 6.23 are generated with ENO3 scheme.

First we show the robustness of this method to occlusions with the synthetic *Sphere* dataset. In this experiment, three central views are used. Because of the occlusions, there is a hole in the reconstructed surface from each central view. After combining these three views together with the proposed algorithm, the holes are all filled in the reconstructed surface. The central views, the reconstructed surfaces from each central view and the final surface reconstruction obtained by combining these three views are shown in Figure 6.21.

Second, we also use the *Sphere* dataset to give an evaluation of the proposed method at different added noise levels. In this experiment, 12 central views and 59 images in total are used. In this group of data, the cameras are located on a hemisphere whose radius is about 65 centimeters. The center of the object (a *sphere*) is at the origin of the world coordinate system with radius 10 centimeters. The evaluations on accuracy and coverage are given in Table 6.3. The accuracy is computed as the average of the distances between the points on the reconstructed surface to the true surface. The coverage is the percentage of the points on the true surface which can find a closest point on the reconstructed surface with the

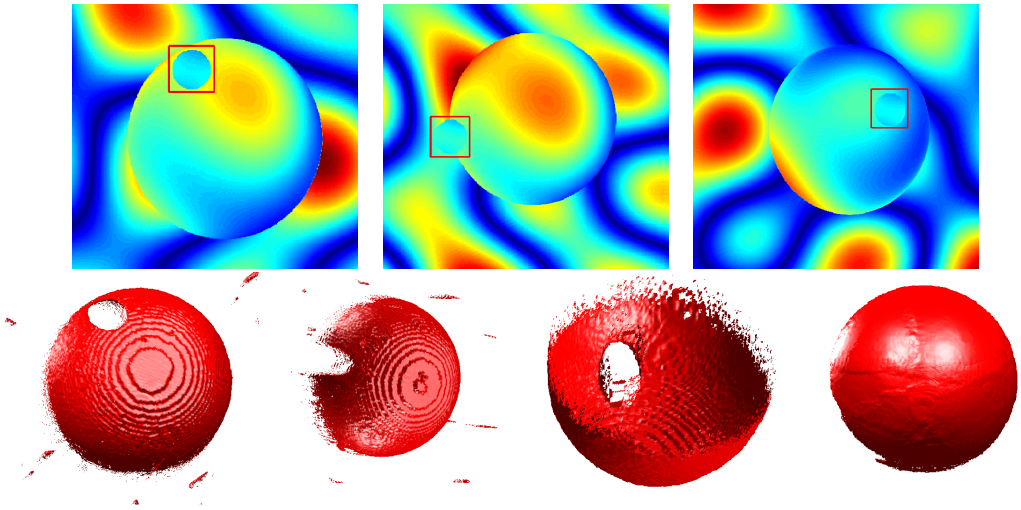


Figure 6.21: The central views and the reconstructed surfaces. Images in the first row are the three central views. The first three images are one view of the reconstructed surface from the corresponding central view. The last image in the second row is one view of the reconstructed surface by combining all these three central views together with the proposed algorithm in this work.

Datasets	Sphere	1% Sphere	3% Sphere	5%bigball
coverage	100%	98.13%	93.73%	83.06%
accuracy	0.09mm	0.235mm	0.372mm	0.9mm
#views	59	59	59	59

Table 6.3: The evaluation on the *Sphere* dataset at different noise levels.

distance between them less than a threshold. In our experiments, the threshold is 0.2 millimeters. The percentages in the first row of the table are the noise levels in the corresponding datasets.

Third, we test the proposed algorithm on the synthetic *Tori* dataset. Four central views, the similar views of the reconstructed 3D model and a reconstruction process are shown in Figure 6.22. The reconstruction process shown in the third row in Figure 6.22 includes one view of the initial surface, one view of the reconstructed surfaces after 10, 30 and 50 iterations. It can be seen that although the initial surface is quite far from the true surface, it can still converge to the true one. We also tested a few other initializations, they all converged to the true

surface. As we pointed out in the previous section, the proposed algorithm is not convex, but it seems that convergence is not a problem by choosing a proper value for the parameter ϵ in Equation (6.27). In all the experiments shown in this chapter, we set it to 2.

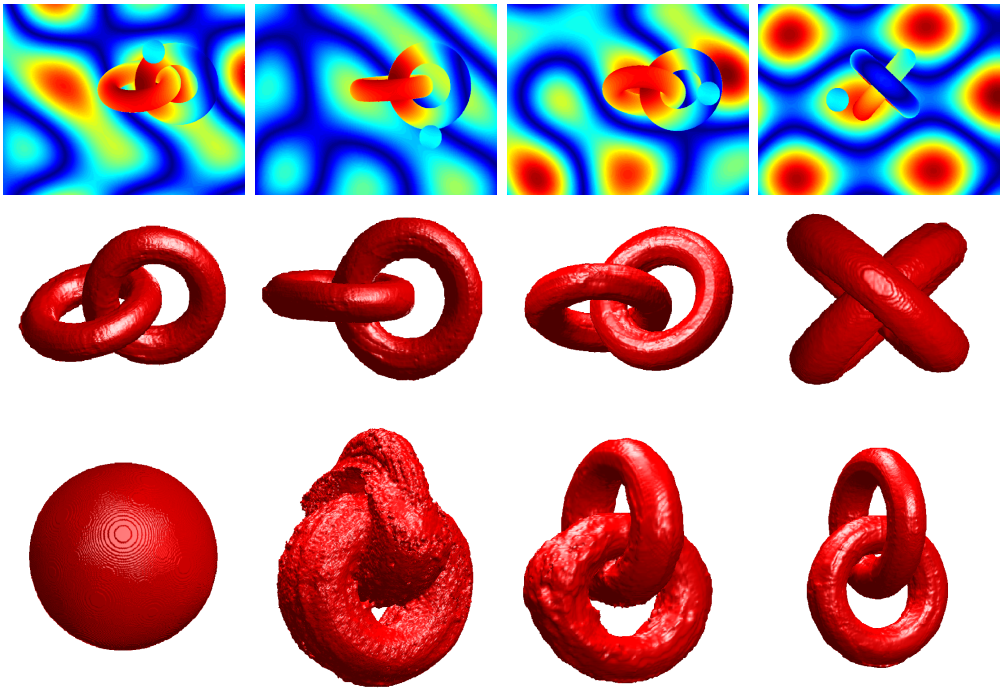


Figure 6.22: The reconstruction of *Tori* dataset. Four central views are shown in the first row and four views of the reconstructed tori are shown in the second row. In the third row four stages of the reconstruction are shown. They are the initial surface, the results after 10, 30 and 50 iterations.

Finally, we test the proposed algorithm on the modified *Dinosaur* dataset. Four input images and four views of the reconstructed 3D model are shown in Figure 6.23.

6.11 Conclusion

In this chapter we have presented two novel frameworks for multiview stereo in the presence of view-varying occlusions and clutter. We cast the problem as re-

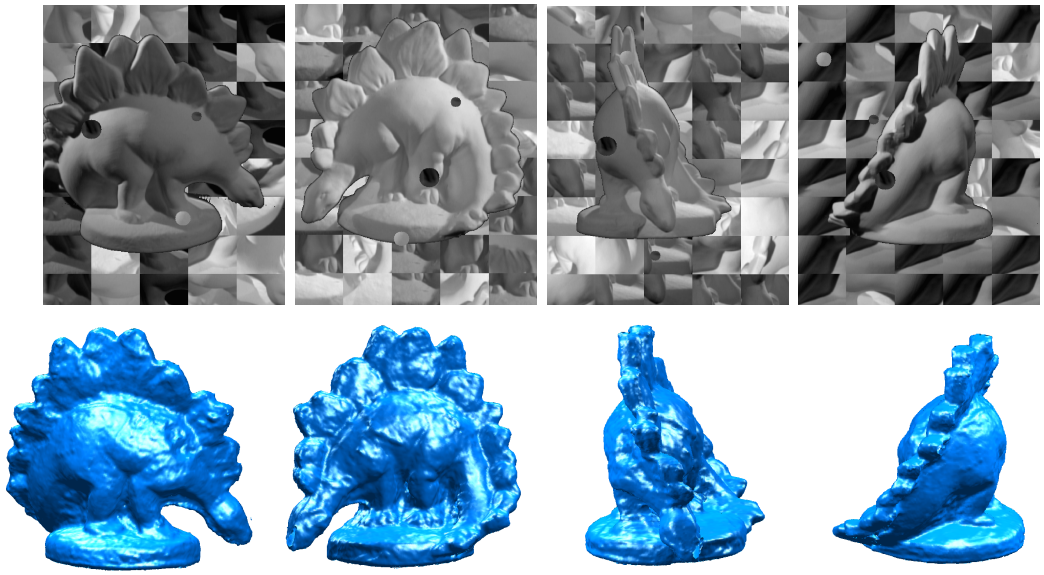


Figure 6.23: The reconstruction of *Dinosaur* dataset. Four central views are shown in the first row and four views of the reconstructed dinosaur are shown in the second row.

covering a smooth surface separating voxels in the scene that are outside objects from voxels that are inside objects via a robust integration of depth map estimates from different vantage points. These frameworks have been designed to be robust to image noise, view-varying occlusions and clutter and to avoid relying on some estimate of the reconstructed object either via the visual hull or silhouettes. The proposed approaches can also be easily modified to take into account novel image formation models or to incorporate general regularization schemes in a globally optimal and computationally efficient numerical implementation. We have illustrated how to robustly perform the integration of all visibilities simultaneously so as to tolerate view-varying occlusions and clutter. Experimental results on synthetic and publicly available real data demonstrate the effectiveness of the proposed method. As an alternative surface integration method for multiview stereo, we take a narrow band around the reconstructed surface from each central view and integrate the narrow bands to reconstruct the whole surface. This is the second method presented in this chapter for solving multiview stereo. By

working on a narrow band of the visibility function, the proposed algorithm is very robust to noise and view-varying occlusions and clutter. Compared with the first method presented in this chapter, this work integrates the visibility functions in a much more computationally efficient way. Although the formulation is not convex, in our experiments, we did not meet any problem for convergence by setting the parameters in the formulation properly.

Chapter 7

Conclusions

7.1 Summary

This Thesis was concerned with the development of algorithms for reconstructing 3D digital models of a scene from a collection of 2D images. Three algorithms were proposed in this work.

The first one is an algorithm for solving multiview stereo with a small baseline based on an off-axis aperture camera model. We can capture images with this camera by rotating the off-axis aperture around the optical axis or by changing the size of the aperture or by altering the distance from the aperture to the lens along the optical axis. When an object is not brought into focus, with the changing of the locations of the aperture, the locations of the projection of the object on the image plane is also changed. But compared with the difference of the locations of the projection, the amount of blur generated by the finite aperture is not negligible. So in this algorithm, as we still use the information generated by stereo, we focus on dealing with blur. To combine the stereo and defocus cues in one formulation, we do not compute the discrepancy between the images taken from different views but compute the discrepancy between the measured image and the image generated from an image formation model. To do

so, we can complete the shape reconstruction and image restoration at the same time. In addition, this imaging device can be used in a small-scale space where the camera motion is constrained by the surrounding environment.

The second algorithm presented in this thesis is a continuous convex formulation for solving multiview stereo with a large baseline. Within this algorithm, a new integration scheme for integrating the visibility functions has been proposed. For each voxel in a certain volume, which contains the objects to be reconstructed, each central view will give a visibility value to it. The integration algorithm allows conflicted visibility values given by different central views and use all the visibility values simultaneously to derive the final visibility value of this voxel. These make this system very robust to time-varying extensive occlusions and clutter and at the same time it does not need to know which cameras share overlapping views. Because of its convexity, this system can guarantee to converge to the global minimum no matter where the initial surface is. The degenerate solution, the null surface, is not included as a global solution.

One shortcoming of the second algorithm is that, with the increasing of the number of central views, the computation load is getting heavy. The third algorithm is designed to solve this problem by integrating the visibility functions within a narrow band around the surface. It keeps the same robustness to occlusions and clutter as the second algorithm, but its computation is much more efficient than the second algorithm.

Our work is based on many algorithms proposed for solving multiview stereo. Chapter 2 gave a detailed review of the multiview stereo methods for dense 3D reconstruction. Based on a taxonomy [146], they are divided into four classes: voxel-based 3D volumetric approaches, partial differential equations (PDEs) based surface evolution methods, the algorithms that compute and merge depth maps and the techniques to reconstruct the 3D surfaces by region growing. The second and third algorithms in our work can be seen as a combination of the first three classes. In these algorithms, we employ the volumetric data representation mim-

icking the approaches of the first class. The object surface is embedded in a 3D volume as the zero level set. The visibility functions of each central view are generated based on the corresponding depth maps. We do not merge the depth maps but the integration of the visibility functions achieves the same purpose, i.e., reconstructing a full scene based on the reconstructions from individual views or local groups. We formulated the reconstruction problem as a energy minimization problem. Staring from a initial setting, the surface is evolved by some partial differential equations and could converge to the solution.

In this thesis, the image formation procedure is reviewed in Chapter 3. In the first algorithm proposed in this work, the measured images are compared with the images computed from a image formation model. The design of the image formation model is one of the central parts of the first algorithm. To test the accuracy and robustness to clutter and occlusions of the second and third algorithms, we generated synthetic data with clutter and occlusions. Image formation procedure is fundamental to each algorithm proposed in this work. Because of the importance of the level set methods to this work, the level set methods and the related numerical schemes were revised in Chapter 4.

In the following chapters, the three algorithms proposed in this work have been described and analyzed in detail and tested with synthetic and real images, and the experimental results and evaluations have been displayed. The qualitative results and the quantitative evaluations show the versatility of the first algorithm and the accuracy and the robustness to occlusions and clutter of the second and third algorithms.

7.2 Main contributions

The three major contributions of this work are the three algorithms we proposed for solving multiview stereo.

Because stereo has more physical dimension than defocus, it is more sensi-

tive to the variation of depth. On the other hand, defocus is more stable than stereo because of its two-dimensionality. In our first algorithm, these two cues are formulated in the same framework naturally. This is the first algorithm in multiview stereo which combines these two cues in this way. The off-axis camera model proposed with this algorithm is a versatile imaging device, which can be used in standard stereo reconstruction, to do reconstruction from defocus and to do reconstruction by combining stereo and defocus. Nobody has modeled a camera in this way before. In the image formation model proposed for this camera, the intensity of each pixel is generated by integrating the radiance from each point on the object surface. Four integrations are involved in the image formation model. Computing the integrations directly is very expensive. We derived a closed form for the computation of the image formation model.

In multiview stereo, most methods are robust to occlusions and there are some methods which have a convex formulation. In our second algorithm, it is the first time to combine the convexity and robustness to occlusions and clutter in one formulation. In this method, we proposed a novel way to map the photoconsistency values along a projection ray to the visibility values. We test several scheme to integrate the visibility functions from each central view into a global visibility function including a novel scheme we proposed in this work. The proposed scheme can tolerate conflicted visibility settings from different central views. This makes this system very robust to occlusions and clutter. The visibility values of a voxel from each central view are treated in the same way and all of them are taken into account to derive the global visibility value of this voxel. This makes this system do not rely on the visual hull or silhouette images at any stage. The convexity of this system guarantees the convergence from any initialization.

The third algorithm is designed to integrate the visibility functions only within a narrow band around the surface. This is a novel formulation in multiview stereo. Compared with the second algorithm the third algorithm is much more efficient in computation. It keeps the same robustness to occlusions and clutter as the

second algorithm and at the same time it improves the robustness to image noise.

7.3 Future Work

There are several aspects of the proposed algorithms that can be extended in the future.

First, for the first algorithm presented in Chapter 6, the *Robust interpolating* function integrates the visibility functions in a heuristic way. A data-dependent criterion instead of a heuristic criterion would be more desirable to integrate the visibility functions. It would be possible to build a data and camera configuration dependent criterion based on probability theory.

Second, for the second method presented in Chapter 6, we built a non-convex system to integrate the visibility functions. Although accurate tuning of the parameters was sufficient to deal with the non-convexity, a convex formulation is still desirable. The vector-valued minimization technique developed in [29, 67] was employed in [30] to modify a non-convex formulation into a convex one. The idea is to consider the optimal constants as functions subject to a constraint on their gradient. It is possible to build a convex formulation based on the vector-valued minimization technique for the second formulation presented in Chapter 6.

Third, a few formulations for solving multiview stereo claim to be convex. However, the convexity for these formulations can only be claimed after the visibility function or the counterparts have been computed. A fully convex formulation for multiview stereo is still not available. This could be another direction to be investigated in the future.

Finally, for the algorithm proposed in Chapter 5, we only tested one scenario with real data, i.e. when the size of the aperture was changed. For testing the algorithm in other scenarios, the calibration of the location of the aperture is involved. This is also work that needs to be done in the future.

Appendix A

The Derivation of the Euler-Lagrange Equation in 1D Case

Here we give the derivation of the Euler-Lagrange equation of single function of single variable. Suppose $\phi = \phi(x)$ is a function with real number x as the independent variable, ϕ' is the derivative of ϕ with respect to x, ϕ and ϕ' are continuous on an interval $[a_0, a_1]$. The function $\Phi(x, \phi, \phi')$ has first and second derivatives with respect to x, ϕ, ϕ'. The functional E is defined as:

$$E[y] = \int_{a_0}^{a_1} \Phi(x, \phi, \phi')dx. \tag{A.1}$$

Suppose functional A.1 attains its extreme when $\phi = \phi_0(x)$, and $\phi(x, \alpha) = \phi_0(x) + \alpha[\phi(x) - \phi_0(x)]$, where α is an arbitrary real number, then $\delta\phi = \phi(x) - \phi(x_0)$ is called a variation of $\phi_0(x)$. The variation of functional A.1 can be defined as:

$$\Psi(\alpha) = \int_{a_0}^{a_1} \Phi(x, \phi_0 + \alpha\delta\phi, \phi_0' + \alpha\delta\phi')dx. \tag{A.2}$$

The derivative of $\Psi(\alpha)$ with respect to α is:

$$\Psi'(\alpha) = \int_{a_0}^{a_1} [\Phi'_{\phi}(x, \phi_0 + \alpha\delta\phi, \phi'_0 + \alpha\delta\phi')\delta\phi + \Phi'_{\phi'}(x, \phi_0 + \alpha\delta\phi, \phi'_0 + \alpha\delta\phi')\delta\phi']dx. \tag{A.3}$$

Then,

$$\Psi'(0) = \int_{a_0}^{a_1} [\Phi'_{\phi}(x, \phi_0, \phi'_0)\delta\phi + F'_{\phi'}(x, \phi_0, \phi'_0)\delta\phi']dx, \tag{A.4}$$

because when $\alpha = 0$, $\Psi(\alpha)$ attain its extreme, so $\Psi'(0) = 0$, i.e.,

$$\int_{a_0}^{a_1} [\Phi'_{\phi}(x, \phi_0, \phi'_0)\delta\phi + \Phi'_{\phi'}(x, \phi_0, \phi'_0)\delta\phi']dx = 0. \tag{A.5}$$

By integrating by parts in eq. A.5 we obtain

$$\int_{a_0}^{a_1} \left[\Phi'_{\phi}(x, \phi_0, \phi'_0) - \frac{d}{dx}\Phi'_{\phi'}(x, \phi_0, \phi'_0)\right] \delta\phi dx = 0. \tag{A.6}$$

Then, according to Theorem 2.3.2

$$\Phi'_{\phi}(x, \phi_0, \phi'_0) = \frac{d}{dx}\Phi'_{\phi'}(x, \phi_0, \phi'_0) \tag{A.7}$$

Equation A.7 is the Euler-Lagrange equation of single function of single variable and it is easy to extend it to multiple dimensions.

Appendix B

Closed-Form Computation of Equation (5.10)

The integration $H_{i,j}(\mathbf{y}_{k,l})$ defined in Section 5.2.2 can be computed in a closed form, which will be presented here. For facilitation to read, the defination of $H_{i,j}(\mathbf{y}_{k,l})$ is repeated here.

$$H_{i,j}(\mathbf{y}_{k,l}) = \int_{k-1/2}^{k+1/2} \int_{l-1/2}^{l+1/2} \int_{i-1/2}^{i+1/2} \int_{j-1/2}^{j+1/2} h(\mathbf{x}, \bar{\mathbf{y}}) d\mathbf{x} d\bar{\mathbf{y}} \tag{B.1}$$

where $h(\mathbf{x}, \bar{\mathbf{y}}) \doteq \frac{1}{2\pi\sigma^2(\mathbf{x})} e^{-\frac{\|\bar{\mathbf{y}} - \gamma^{-1}\pi[P(\mathbf{x})]\|^2}{2\sigma^2(\mathbf{x})}}$. According to the description in Section 5.2.2, $\sigma(x)$ is the same everywhere in the small square, so it can be denoted by a constant σ. $\pi[\mathbf{P}(\mathbf{x})]$ is the vector including the first two coordinates of the projection of a point $(x_1, x_2, x_3)^T$ on the image plane. Because we suppose that the small square centered at point $(x_1, x_2, x_3)^T$ is parallel to the image plane, so x_3 is a constant within this small square. According to Equation (5.5), $\pi[\mathbf{P}(\mathbf{x})] = (Ax_1 + B_1, Ax_2 + B_2)^T$ with $A = -\gamma^{-1}[(1 - v/v_0)C_3/(x_3 - C_3) + v/x_3]$, $B_1 = \gamma^{-1}(1-v/v_0)C_1x_3/(x_3-C_3)$ and $B_2 = \gamma^{-1}(1-v/v_0)C_2x_3/(x_3-C_3)$. Now $H_{i,j}(\mathbf{y}_{k,l})$

can be expressed by the following expression:

$$H_{i,j}(\mathbf{y}_{k,l}) = \frac{1}{2\pi\sigma^2}\int_{k-\frac{1}{2}}^{k+\frac{1}{2}}\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} e^{-\frac{(\bar{y}_1-Ax_1-B_1)^2}{2\sigma^2}}dx_1 d\bar{y}_1 \int_{l-\frac{1}{2}}^{l+\frac{1}{2}}\int_{j-\frac{1}{2}}^{j+\frac{1}{2}} e^{-\frac{(\bar{y}_2-Ax_2-B_2)^2}{2\sigma^2}}dx_2 d\bar{y}_2 \tag{B.2}$$

We know that the Error function is defined as

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-\frac{t^2}{2}}dt \tag{B.3}$$

By changing variables, we have the following equation

$$\frac{2}{\sqrt{\pi}}\int_{i-\frac{1}{2}}^{i+\frac{1}{2}} e^{-\frac{(\bar{y}_1-Ax_1-B_1)^2}{2\sigma^2}}dx_1 = \frac{\sigma}{A}\left[erf(I_{p0}) - erf(I_{q0})\right] \tag{B.4}$$

where $I_{p0} = \frac{\bar{y}_1-A(i-\frac{1}{2})-B_1}{\sigma}$ and $I_{q0} = \frac{\bar{y}_1-A(i+\frac{1}{2})-B_1}{\sigma}$ with A, B_1 and σ having the same meaning as in Equation (B.2).

Define the following function

$$Exp\_(x) = \frac{1}{\sqrt{\pi}}e^{-x^2} + x \cdot erf(x) \tag{B.5}$$

One property of function $Exp\_(x)$ is that its derivative is the Error function $erf(x)$, so by combining Equations (B.2), (B.3), (B.4) and (B.5), we can have the following equation

$$H_{i,j}(\mathbf{y}_{k,l}) = \frac{\sigma^2}{8A^2}E_1 \cdot E_2 \tag{B.6}$$

where $E_i = Exp\_I_{p_i} - Exp\_I_{q_i} - Exp\_I_{r_i} + Exp\_I_{s_i}$, with the included variables defined in the following expressions:

$$\begin{aligned} I_{p_1} &= k + \frac{1}{2} - A(i - \frac{1}{2}) - B_1 \\ I_{q_1} &= k - \frac{1}{2} - A(i - \frac{1}{2}) - B_1 \\ I_{r_1} &= k + \frac{1}{2} - A(i + \frac{1}{2}) - B_1 \end{aligned}$$

$$
\begin{aligned}
I_{s_1} &= k - \frac{1}{2} - A(i + \frac{1}{2}) - B_1 \\
I_{p_2} &= l + \frac{1}{2} - A(j - \frac{1}{2}) - B_2 \\
I_{q_1} &= l - \frac{1}{2} - A(j - \frac{1}{2}) - B_2 \\
I_{r_1} &= l + \frac{1}{2} - A(j + \frac{1}{2}) - B_2 \\
I_{s_1} &= l - \frac{1}{2} - A(j + \frac{1}{2}) - B_2
\end{aligned} \tag{B.7}
$$

Equation (B.6) gives a closed form for computing the energy given to a pixel by a small square represented by a sampled point on the object surface.

References

[1] http://vision.middlebury.edu/. 143

[2] http://www.vision.caltech.edu/bouguetj/calib_doc. 53

[3] S. Abraham and T. Hau. Towards autonomous high-precision calibration of digital cameras. *Proceedings of SPIE Videometrics V, El-Hakim(Ed.)*, vol. 3174:82–93, 1997. 53

[4] D. Adalsteinsson and J. Senthian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995. 22

[5] A. Agarwal and B. Triggs. Learning to track 3D human motion from silhouettes. *In International Conference on Machine Learning*, pages 9–16, 2004. 15

[6] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic view. *IEEE Trans*, 11(2):137–149, 1989. 14

[7] B. Appleton and H. Talbot. Globally minimal surface by continuous maximal flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:106–118, 2006. 123

[8] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *Int. Journal of Computer Vision*, Vol. 1(2):107–131, 1987. 30

[9] N. Ayache and F. Lustman. Trinocular stereo vision for robotics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 13:73–85, 1991. 30

[10] H. Baker and T. Binford. Depth from edge and intensity based stereo. *In Proc 7th Int. Joint Conf. Artificial Intelligence*, pages 631–636, 1981. 30

[11] S. Baker and I. Matthews. Lucas-kanade 20 years on: a unifying framework. *IJCV*, 56(3):221–255, 2004. 35

[12] B. G. Baumgart. Geometric modelling for computer vision. *PhD thesis, CS Dept, Stanford U.*, October 1974. 10, 14

[13] P. R. Beaudet. Rotationally invariant image operators. *In Proceedings of the International Joint Conference on Pattern Recognition*, pages 579–583, 1978. 33

[14] R. Bhotika, D. Fleet, and K. Kutulakos. A probabilistic theory of occupancy and emptiness. *ECCV*, vol. 3:pages 112–132, 2002. 18

[15] J. Bittner and P. Wonka. Visibility in computer graphics. *Environment and Planning B: Planning and Design*, 30(5):729–756, 2003. 12

[16] M. Black and P. Anandan. A framework for the robust estimation of optical flow. *ICCV*, pages 231–236, 1993. 30

[17] M. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, vol. 19(1):57–91, 1996. 30, 31

[18] A. Bobick and S. Intille. Large occlusion stereo. *IJCV*, vol. 33(3):181–200, 1999. 31

[19] R. Bodo, K. Uwe, A. Smith, B. Thomas, K. Reinhard, and S. Hans-Peter. A silhouette based human motion tracking system. *Technical Report, CITR, The University of Auckland, New Zealand*, ISSN: 1178-3581, 2005. 15

[20] T. Bonfort and P. Sturm. Voxel carving for specular surface. *ICCV*, pages pages 691–696, 2003. 113

[21] M. Born and E. Wolf. *Principles of optics.* (Pergamon, Oxford, 1980). 45

[22] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006. 13

[23] Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. *Medical Image Computing and Computer-Assisted Intervention*, pages 276–286, 2000. 18

[24] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. *ICCV*, pages 26–33, 2003. 20

[25] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. *CRPR*, pages 648–655, 1998. 18

[26] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, Vol. 23(11):1222–1239, 2001. 18, 31

[27] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multiview reconstruction using robust binocular stereo and surface meshing. *CVPR*, pages pages 1–8, 2008. 10

[28] M. Brand, K. Kang, and D. B. Cooper. An algebraic solution to visual hull. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004. 10

[29] E. S. Brown, T. F. Chan, and X. Bresson. A convex relaxation method for a class of vector-valued minimization problems with applications to mumford-shah segmentation. 2010. 164

[30] E. S. Brown, T. F. Chan, and X. Bresson. Globally convex Chan-Vese image segmentation. 2010. 164

[31] T. Chan and L. Vese. Active contours without edges. *IEEE Trans. on Image Processing*, 10:266–277, 2001. 7, 60, 78, 117, 148

[32] T. Chan and L. Vese. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, vol. 50(3):271–293, 2002. 60, 78

[33] S. Chaudhuri and A. Rajagopalan. *Depth from defocus: a real aperture imaging approach*. Springer Verlag, 1999. 90, 96

[34] G. K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. *CVPR*, 2:714–720, 2000. 14, 15

[35] A. J. Chorin. Flame advection and propagation algorithms. *Journal of Computational Physics*, 35:1-11, 1980. 22

[36] T. Clarke and J. Fryer. The development of camera calibration methods and models. *The Photogrammetric Record*, vol. 16(91):51–66, 1998. 53

[37] D. Cohen-Or, Y. Chrysanthou, C. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Trans. on Visualization and Computer Graphics*, 2002. 12

[38] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. *Computer Vision and Computer Graphics*, pages 25–47, 2000. 26

[39] B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. *In International Workshop on Vision Algorithms*, pages 100–114, 1999. 11

[40] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *In Proceedings of SIGGRAPH*, pages pages 303–312, 1996. 13, 28, 32, 117

[41] R. DeBar. Fundamentals of the KRAKEN code. *Technical Report UCIR-760,LLNL*, 1974. 22

[42] F. Devernay and O. Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, vol. 13, no. 1:14–24, 2001. 56

[43] Q. Dou and P. Favaro. Off-axis aperture camera: 3D shape reconstruction and image restoration. *CVPR*, pages 1–7, 2008. 90

[44] Q. Dou and P. Favaro. A convex multi-view stereo formulation with robustness to occlusions and time-varying clutter. *ICVGIP*, pages 243–250, 2010. 113

[45] L. Dreschler and H. Nagel. Volumetric model and 3D-trajectory of a moving car derived from monocular TV-frame sequence of a street scene. *In Computer Graphics and Image Processing*, vol. 20:199–228, 1982. 33

[46] F. Durand. 3D visibility: analytical study and applications. *Ph.D. thesis, Universite Joseph Fourier, Grenoble, France*, 1999. 12

[47] P. Eisert, E. Steinbach, and B. Girod. Multi-hypothesis, volumetric reconstruction of 3d objects from multiple calibrated camera views. *In ICASSP 99*, pages 3509–3512, 1999. 18

[48] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *In Proceedings of SIGGRAPH*, pages 736–744, 2002. 22

[49] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modelling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004. 10, 12, 15, 21, 26

[50] J. Q. Fang and T. S. Huang. A corner finding algorithm for image analysis and registration. *In Proceedings of AAAI Conference*, pages 46–49, 1982. 33

[51] O. Faugeras and R. Keriven. Complete dense stereovision using level set methods. *ECCV*, pages 379–393, 1998. 9, 21, 30

[52] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Trans. on Image Processing*, Vol. 7(3):336–344, 1998. 10, 21, 22, 26

[53] P. Favaro and S. Soatto. *3-D shape reconstruction and image restoration: exploiting defocus and motion-blur*. Springer Verlag, 2006. 90, 96

[54] L. Ford and D. Fulkerson. *Flows in networks*. Princeton University Press, 1962. 19

[55] J.-S. Franco and E. Boyer. Exact polyhedral visual hulls. *In Fourteenth British Machine Vision Conference*, pages 329–338, 2003. 10

[56] T. Fromherz and M. Bichsel. Sahpe from multiple cues: Integrating local brightness information. *In ICYCS*, 1995. 18

[57] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, vol. 6:35–49, 1993. 31

[58] P. Fua. From multiple stereo views to multiple 3D surfaces. *IJCV*, 24(1):19–35, 1997. 5, 31

[59] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. *ECCV*, Vol. 1:564–577, 2006. 15

[60] Y. Furukawa and J. Ponce. Higu-fidelity image-based modelling. *Technical Report, UIUC*, 2006. 26

[61] Y. Furukawa and J. Ponce. Accurate, dense and robust multi-view stereopsis. *CVPR*, 2007. 5, 10, 34, 35, 36

[62] Y. Furukawa and J. Ponce. Dense 3D motion capture from synchronized video streams. *CVPR*, page pages, 2008. 113, 114

[63] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, Vol. 12:16–22, 2000. 30

[64] P. Gargallo, E. Prados, and P. Sturm. Minimizing the reprojection error in surface reconstruction from images. *ICCV*, 2007. 10, 21

[65] M. Goesele, B. Curless, and S. Seitz. Multi-view stereo revisited. *CVPR*, 2006. 10, 32, 117

[66] M. Goesele, N. snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. *ICCV*, 2007. 5, 10

[67] T. Goldstein, X. Bresson, and S. Osher. Global minimization of markov random fields with applications to optical flow. *UCLACAM Report 09-77*, September, 2009. 164

[68] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. *SIGGRAPH*, pages 43–54, 1996. 15

[69] C. Grant. Visibility algorithms in image synthesis. *Ph.D. thesis, U. of Califonia, Davis*, 1992. 12

[70] J. F. Greenleaf, T. S. Tu, and E. H. Wood. Computer generated 3D oscilloscopic images and associated techniques for display and study of the spacial distribution of pulmonary blood flow. *IEEE Transactions on Nuclear Science*, 17(3):353–359, 1970. 10

[71] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society Series B*, 51:271–279, 1989. 18

[72] M. Habbecke and L. Kobbelt. Iterative multi-view plane fitting. *In Proc. of VMV*, pages 73–80, 2006. 34

[73] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. *CVPR*, 2007. 10, 34, 35, 36

[74] R. Haberman. *Elementary applied partial differential equations*. Prentice-Hall, Englewood, Clliffs, New Jersey, 1983. 67

[75] C. Harris and M. Stephens. A combined corner and edge detector. *In Alvey Vision Conference*, pages 147–151, 1988. 33

[76] S. W. Hasinoff and K. N. Kutulakos. Confocal stereo. *European Conf. on Computer Vision*, pages 620–634, 2006. 90

[77] X. He, K. E. Torrance, F. X. Sillion, and D. P. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics*, vol. 25(4):175–186, 1991. 40

[78] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. *CVPR*, pages 1106–1112, 1997. 56

[79] C. Hernandez, G. Vogiatzis, and R. Cipolla. Probabilistic visibility for multi-view stereo. *CVPR*, page pages, 2007. 10, 13, 32

[80] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. *CVPR*, 2009. 113

[81] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. *ECCV*, pages pages 117–126, 1996. 28, 29

[82] C. W. Hirt and B. D. Nichols. A computational method for free surface hydrodynamics. *Journal of Pressure Vessel Technology*, pages 103:136–140, 1981. 22

[83] A. Hornung and L. Kobbelt. Hierarchical volumetric multiview stereo reconstruction of manifold surfaces based on dual graph embedding. *CVPR*, pages 503–510, 2006. 10, 11, 15, 18

[84] H. Ishikawa and D. Geiger. Occlusions, discontinuities and epipolar lines in stereo. *ECCV*, pages 232–248, 1998. 18

[85] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys. 126*, pages 202–228, 1996. 73

[86] H. Jin, D. Cremers, A. Yezzi, and S. Soatto. Shedding light on stereoscopic segmentation. *CVPR*, Vol. 1:36–42, 2004. 113

[87] J. Jost and X. Li-Jost. *Calculus of variations*. UK: Cambridge University Press, 1998. 24, 79

[88] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16:920–932, 1994. 30

[89] R. Keriven. A variational framework to shape from contours. *Tech. Rep. 2002-221*, 2002. 26

[90] R. Keriven and O. Faugeras. Variational principles, surface evolution, pdes, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, 1998. 13

[91] R. Koch, M. Pollefeys, and L. V. Gool. Multi viewpoint stereo from uncalibrated video sequences. *ECCV*, volume I:55–71, 1998. 31

[92] R. Koch, M. Pollefeys, and L. V. Gool. Robust calibration and 3D geometric modeling from large collections of uncalibrated images. *DAGM*, pages 413–420, 1999. 31

[93] K. Kolev, M. Klodt, T. Brox, and D. Cremers. Propagated photoconsistency and convexity in variational multiview 3D reconstruction. *In Workshop on Photometric Analysis for Computer Vision*, 2007. 10, 13, 15, 26, 27

[94] K. Kolev, M. Klodt, T. Brox, and D. Cremers. Continuous global optimization in multiview 3D reconstruction. *International Journal of Compuer Vision*, 84:80–96, 2009. viii, 5, 120, 121

[95] V. Kolmogorov and R. Zabih. Visual correspondence with occlusions using grapg cuts. *ICCV*, pages 508–515, 2001. 18

[96] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. *ECCV*, Volume III:82–96, 2002. 19

[97] K. Kutulakos. Approximate n-view stereo. *ICCV*, vol. I:67–83, 2000. 18

[98] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *Int. Journal of Computer Vision*, Vol. 38(3):199–218, 2000. 9, 10, 16, 17, 18

[99] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. *ICCV*, 2007. 32

[100] J.-O. Lachaud and B. Taton. Deformable model with adaptive mesh and automated topology changes. *In Proceedings of the Fourth International Conference on 3-D Digital Imaging ang Modeling*, 2003. 22

[101] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti. Modelling merging and fragmentation in multiphase flows with SURFER. *Journal of Computational Physics*, 113:134-147, 1994. 22

[102] A. Laurentini. The visual hull: a new tool for contour-based image understanding. *Proc. 7th Scandinavian Conf. Image Analysis*, pages 993–1002, 1991. 10

[103] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. 14

[104] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995. 14

[105] A. Laurentini. How many 3D shapes can be understood from 2D silhouettes. *Computer Vision and Image Understanding*, 67(1):81–87, 1997. 14

[106] V. Lempitsky, Y. Boykov, and D. Ivanov. Oriented visibility for multiview reconstruction. *ECCV*, vol. 3953 of LNCS:pages 226–238, 2006. 11, 18

[107] R. K. Lenz and R. Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-D machine vision metrology. *IEEE Trans. on PAMI*, vol. 10(5):713–720, 1988. 56

[108] M. Levoy, R. Ng, A. Adams, M. Footer, and M. Horowitz. Light field microscopy. *ACM SIGGRAPH*, pages 924–934, 2006. 90

[109] M. Lhuiller and L. Quan. Surface reconstruction by integrating 3D and 2D data of multiple views. *ICCV*, pages 1313–1320, 2003. 26

[110] M. Lhuillier and L. Quan. Match propogation for image-based modeling and rendering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1140–1146, 2002. 34

[111] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *In Proceedings of the SIGGRAPH'87 Conference*, Vol. 21:163–170, 1987. 73, 75

[112] D. Lowe. Distinctive image features from scale-invariant keypopints. *International Journal of Computer Vision*, Vol. 2(60):91–110, 2004. 34

[113] D. G. Lowe. Object recognition from local scale-invariant features. *In International Conference on Computer Vision*, pages 1150–1157, 1999. 32

[114] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, Vol. 194:283–287, 1976. 29, 30

[115] S. R. Marschner, S. H. Westin, E. P. F. Lafortune, and K. E. Torrance. Image-based bidirectional reflectance distribution function measurement. *Applied Optics*, vol. 39(16):2592–2600, 2000. 40

[116] W. N. Martin and J. K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983. 10, 14

[117] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. *ACM SIGGRAPH*, Special Interest Group on Computer Graphics:369–374, 2000. 14

[118] E. A. Maxwell. *The methods of plane projective geometry based on the use of general homogeneous coordinates*. Cambridge University Press, 1964 reprint of 2nd edition, 1946. 48

[119] T. McInerney and D. Terzopoulos. T-snakes: topology-adaptive sankes. *Medical Images Analysis*, 4(2):pages 73–91, 2000. 21

[120] P. Merrell, A. Akbarzadeh, L. wang, P. Mordohai, J.-M.Frahm, R.Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. *ICCV*, 2007. 10, 121

[121] S. Moezzi, L.-C. Tai, and P. Gerard. Virtual view generation for 3D digital video. *IEEE Multimedia*, Vol. 4(1):18–26, 1997. 15

[122] H. Moravec. Visual mapping by a robot rover. *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 598–600, 1979. 34

[123] H. Moravec. Rover visual obstacle avoidance. *In International Joint Conference on Artificial Intelligence*, pages 785–790, 1981. 32

[124] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. Pure Appl. Math*, vol. 42:577–685, 1989. 60

[125] T. Nanjo. Stereoscopic retinal camera including vertically symmetrical apertures. *Patent N.5302988*, 1994. 90

[126] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. *Technical Report NBS-MN-160, National Bureau Standards, Washington, DC, USA.*, 1977. 40

[127] W. Niem. Error analysis for silhouette-based 3D shape estimation from multiple views. *Workshop on Synthetic-Natural Hybrid Coding and Three-dimensional Imaging*, 1997. 14

[128] H. Noborio, S. Fukada, and S. Arimoto. Contribution of the octree approximating three-dimensional objects by using multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):769–782, 1988. 14, 15

[129] W. Noh and P. Woodward. A simple line interface calculation. *In Proceedings of Fifth International Conference on Fluid Dynamics*, 1976. 22

[130] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *In IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 7:139–154, 1985. 30

[131] S. Osher and R. Fedkiw. *The level set methods and dynamic implicit surfaces*. Springer Verlag, 2002. 22, 66

[132] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, Vol. 79:12–49, 1988. 6, 22, 60, 62, 63

[133] G. P. Otto and T. K. W. Chau. Region-growing algorithm for matching of terrain images. *Image Vision Comput.*, 7(2):83–94, 1989. 34

[134] D. Peng, B. Merriman, S. Osher, H.-K. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999. 65

[135] J.-P. Pons and J.-D. Boissonnat. Delaunay deformable models: topology-adaptive meshes based on the restricted Delaunay triangulation. *CVPR*, pages 1–8, 2007. 22

[136] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. *CVPR*, Volume II:822–827, 2005. 10, 21, 26

[137] M. Potmesil. Generating octree models of 3D objects from their silhouettes in sequences of images. *Computer Vision, Graphics and Image Processing*, 40:1–20, 1987. 14

[138] S. Roy and I. Cox.A. Maximum-flow formulation of the n-camera stereo correspondence problem. *ICCV*, pages 735–743, 1998. 18

[139] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. *CVPR*, vol. 2:pages 49–54, 1999. 18

[140] Y. Sato, M. D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. *In Proc. of ACM SIGGRAPH*, pages 75–84, 2000. 40

[141] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *IJCV*, vol. 28(2):155–174, 1998. 31

[142] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47:7–42, 2002. 31, 90

[143] C. Schmid and R. Mohr. Combining gray-value invariants with local constraints for object recognition. *CVPR*, pages 872–877, 1996. 34

[144] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):530–534, 1997. 32

[145] S. Seitz and C. Dyer. Photorealistic scene reconstructon by voxel coloring. *CVPR*, pages 1067–1073, 1997. 10, 11, 18

[146] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, 1(2):519–528, 2006. 2, 9, 10, 161

[147] J. A. Sethian. *Level set mothods and fast marching methods.* Cambridge University Press, 2nd edition, 1999. 66

[148] F. X. Sillion and C. Puech. *Radiosity and global illumination*. Morgan Kaufmann Publisherrs, 1994. 40, 41

[149] A. Singh and M. Shneier. Gray-level corner detection a generalization and a robust real time implementation. *In Proceedings of the Computer Vision Graphics Image Processing*, Vol. 51:54–69, 1990. 33

[150] S. N. Sinha, P. Mordohai, and M. Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. *ICCV*, 2007. 10

[151] G. Slabaugh, B. Culbertson, T. Malzbender, and M. Stevens. Methods for volumetric reconstruction of visual scenes. *IJCV*, 57(3):pages 179–199, 2004. 18

[152] A. R. Smith and J. F. Blinn. Blue screen matting. *SIGGRAPH*, pages 259–268, 1996. 16

[153] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. *CVPR*, Volume I:345–352, 2000. 14, 15

[154] S. Soatto, A. Yezzi, and H. Jin. Tales of shape and radiance in multiview stereo. *ICCV*, pages 974–981, 2003. 26, 27

[155] M. Sormann, C. Zach, J. Bauer, K. Karner, and H. Bischof. Watertight multi-view reconstruction based on volumetric graph-cuts. *SCIA*, 2007. 10

[156] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):pages 344–358, 1995. 28

[157] S. K. Srivastava and N. Ahuja. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics and Image Processing*, 49:68–84, 1990. 15

[158] J. Starck, G. Miller, and A. Hilton. Volumetric stereo with silhouette and feature constraints. *BMVC*, Vol. 3:1189–1198, 2006. 15, 32

[159] A. Starti and S. Tubaro. Image-based multiresolution implicit object modeling. *EURASIP*, Vol. 1:1053–1066, 2002. 9

[160] C. Strecha, R. Fransens, and L. V. Gool. Combined depth and outlier estimation in multi-view stereo. *CVPR*, 2006. 5, 10

[161] C. Strecha, R. Tuytelaars, and L. V. Gool. Dense matching of multiple wide-baseline views. *ICCV*, pages 1194–1201, 2002. 32

[162] J. C. Strikwerda. *Finite difference schemes and partial differential equations*. Wadsworth and Brooks/Cole Advanced Books and Software, Pacific Grove, California, 1989. 69

[163] J. Strutz. 3D endoscopy. *In HNO*, vol. 41:128–130, 1993. 90

[164] M. V. Sudhamani and C. R. Venugopal. Segmentation of images through clustering to extract color features: an application for image retrieval. *International Journal of Computer Science*, vol. 2, No. 1:54–61, 2007. 13

[165] S. Sullivan and J. Ponce. Automatic model construction, pose estimation and object recognition from photographs using triangular splines. *ICCV*, pages pages 510–516, 1998. 9

[166] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994. 64, 65

[167] I. Sutherland, R. Sproull, and R. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys*, vol. 6:1–55, 1974. 12

[168] R. Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing*, 58(1):23–32, 1993. 15

[169] R. Szeliski. A multi-view approach to motion and stereo. *CVPR*, 1999. 10

[170] S. Tran and L. Davis. 3D surface reconstruction using graph cuts with surface constraints. *ECCV*, Vol. 2:219–231, 2006. 10, 15

[171] A. Treuille, A. Hertzmann, and S.Seitz. Example-based stereo with general BRDFs. *ECCV*, pages pages 457–469, 2004. 113

[172] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision*. N. J.: Prentice Hall, 1998. 43, 56

[173] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Int. Journal Robotics and Automation*, vol. 3(4):323–344, 1987. 53, 56

[174] G. Turk and M. Levoy. Zippered polygon meshes from range images. *In SIGGRAPH*, pages 311–318, 1994. 28

[175] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *FnT Comp. Graphics and Vision*, pages 177–280, 2008. 33

[176] R. Vaillant and O. D. Faugeras. Using extremal boundaries for 3d object modelling. *IEEE Trans. Pattern Analysis and Machina Intelligence*, 14:157–173, 1992. 9

[177] G. Vogiatzis, C. Hernandez, P. H. S. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29:2241–2246, 2007. 10, 11

[178] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. *CVPR*, Vol. 1:391–398, 2005. 10, 11, 13, 18, 20, 123

[179] G. Vogiatzis, P. H. S. Torr, S. M. Seitz, and R. Cipolla. Reconstructing relief surfaces. *BMVC*, pages 117–126, 2004. 20

[180] G. J. Ward. Measuring and modeling anisotropic reflection. *In Proc. of ACM SIGGRAPH*, pages 265–272, 1992. 40

[181] M. Watanabe and S. K. Nayar. Telecentric optics for computational vision. pages 439–451, 1996. 99

[182] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. on PAMI*, vol. 14(10):965–980, 1992. 56

[183] M. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. *ICCV*, pages 917–924, 1998. 28

[184] A. Wiley and K. Wong. Geometric calibration of zoom lenses for computer vision metrology. *PE&RS*, vol. 61(1):69–74, 1995. 53

[185] J. Xu, O. Chutatape, C. Zheng, and P. Kuan. Three dimensional optic disc visualization from stereo images via dual registration and ocular media optical correction. *British Journal of Ophthalmology*, 90:182–185, 2006. 90

[186] A. Yezzi, G. Slabaugh, R. Cipolla, and R. Schafer. A surface evolution approach of probabilistic space carving. *First International Symposium on 3D Data Processing Visualization and Transmission*, pages 618–621, 2002. 9

[187] A. Yezzi and S. Soatto. Stereoscopic segmentation. *Int. Journal of Computer Vision*, 53(1):31–43, 2003. 26

[188] T. Yu, N. Ahuja, and W.-C. Chen. SDG cut: 3D reconstruction of non-lambertian objects using graph cuts on surface distance grid. *CVPR*, Vol. 2:2269–2276, 2006. 15

[189] C. Zach, T. Pock, and H. Bishop. A globally optimal algorithm for robust TV-L1 range image integration. *ICCV*, 2007. 5, 10, 121

[190] A. Zaharescu, E. Boyer, and R. Horaud. Transformesh: A topology-adaptive mesh-based approach to surface evolution. *In ACCV*, 2007. 10, 21

[191] G. Zeng, S. Paris, L. Quan, and F. Sillion. Progressive surface reconstruction from images using local prior. *ICCV*, pages 1230–1237, 2005. 18

[192] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. on PAMI*, vol. 22(11):1330–1334, 2000. 53

[193] Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geomery. *Artificial Intelligence*, Vol. 78:87–119, 1995. 32

[194] Z. Zhang and Y. Shan. Progressive scheme for stereo matching. *In SMILE'00 Workshop on 3D Struct. from Mult. Images of Large-Scale Environments*, pages 68–85, 2001. 34

[195] Z. Zhang, H. Wang, and E. Teoh. Analysis of gray level corner detection. *Pattern Recognition Letters*, Vol. 20:149–162, 1999. 34

[196] A. Zomet and S. K. Nayar. Lensless imaging with a controllable aperture. *Computer Vision and Pattern Recognition*, 1(1063-6919):339–346, 2006. 91

[197] O. A. Zuniga and R. Haralick. Corner detection using the facet model. *In Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 30–37, 1983. 33